Colección
Recursos Informáticos

VBA Excel 2007

Programar en Excel: Macros y Lenguaje VBA

Michèle AMELOT

Con la colaboración de
Guillermo SOM ("El Guille")

Microsoft Most Valuable Professional

Puede descargar los ficheros www.ediciones-eni.com

INFORMÁTICA TÉCNICA

eni ediciones

Copyrighted material

Ediciones ENI: novedades cada mes a la venta en su librería habitual

Certificaciones

Colección Certificaciones

Esta colección tiene como objetivo ayudarle en la preparación de diferentes certificaciones, cubriendo todos los objetivos oficiales.

500 a 700 pág. - 17,8 x 21,6 cm

Precio venta público: 52,80 € cada título

400 a 500 pág. - 17,8 x 21,6 cm

Precio venta público: 42 € cada título

Manuales Técnicos

Colección Recursos Informáticos

Redes, sistemas, programación, SGBDR, tecnologías Internet o Intranet, informática industrial... Redactados por formadores profesionales y consultores, los libros de esta colección le aportarán una ayuda eficaz para descubrir y aplicar estas diferentes técnicas.

350/450 pág. - 17,8 x 21,6 cm

Precio venta público: 26.95 € cada título

Diseño y Multimedia

Colección Studio Factory

Cada libro de esta colección presenta de forma clara y ampliamente ilustrada el conjunto de las funciones del programa, constituyendo así un verdadero manual de referencia.

250/350 pág. - 17,8 x 21,6 cm.

Precio venta público: 14.96 € cada título

Colección Technote

Rápido y conciso, para saber todo sobre un tema técnico.

120 a 150 pág. - 17,8 x 21,6 cm

Precio venta público: 9.00 € cada título

Manuales Prácticos

Colección Ofimática Profesional

Basándose en explicaciones claras y detalladas, estos manuales permiten dominar completamente sus aplicaciones informáticas.
Público principiante o avanzado.

290 a 330 páginas - 17,8 x 21,6 cm

Precio venta público: 13.99 € cada título

Colección Pasaporte

Todas las funciones extensamente ilustradas con la ayuda de cuadros de diálogo y de dibujos de pantalla. Las funciones están agrupadas por temas, lo que le permitirá encontrar fácilmente la información deseada.

250 pág. - 14,85 x 21 cm.

Precio venta público: 11.96 € cada título

Ofimática

Colección Zona Principiante

Esta colección está destinada a iniciarle en la informática de la forma más clara posible.
Con ella tendrá todas las claves, herramientas y trucos para el uso cotidiano de su PC.

150 pág. - 17,8 x 21,6 cm

Precio venta público: 5.95 € cada título

Autoformación

Colección Sigue el Ejemplo

Una colección basada en numerosos ejemplos paso a paso, donde explicaciones teóricas y prácticas se asocian para ofrecerle una formación dinámica, concreta y completa.

300 a 400 pág. - 17,8 x 21,6 cm.
Con disquete de ejercicios.

Precio venta público: 23.44 € cada título

Guías Prácticas

Colección Triunfar Con

Una excelente guía práctica para tener siempre al alcance de su PC, que le permitirá encontrar en unos segundos la función deseada o recordar en un vistazo la operación que se ha de realizar.

110 pág. - 14,85 x 21 cm

Precio venta público: 5.98 € cada título

Guías por Niveles

Colección Clave

Una colección dirigida a un público principiante, donde se puede encontrar toda la información sobre el programa que se estudia. Para algunos títulos (Word, Excel), se han desarrollado dos niveles, que facilitan el aprendizaje por módulos: funciones básicas y funciones avanzadas.

150 a 160 pág. - 17,8 x 21,6 cm

Precio venta público: 7.78 € cada título

Copyrighted material

VBA Excel 2007

Programar en Excel: Macros y Lenguaje VBA

This one
EN4Y-DBF-2SAR
Copyrighted material

Todas las marcas citadas han sido registradas por su respectivo editor.

Reservados todos los derechos. El contenido de esta obra está protegido por la ley, que establece penas de prisión y/o multas, además de las correspondientes indemnizaciones por daños y perjuicios, para quienes reprodujeren, plagiaren, distribuyeren o comunicaren públicamente, en todo o en parte, una obra literaria, artística o científica, o su transformación, interpretación o ejecución artística fijada en cualquier tipo de soporte o comunicada a través de cualquier medio, sin la preceptiva autorización.

Copyright - Editions ENI - Diciembre 2007
ISSN: 1629-7458
ISBN: 978-2-7460-3915-5
Edición original: 978-2-7460-3780-9

Ediciones ENI es una marca comercial registrada de Ediciones Software.

Ediciones ENI

P° Ferrocarriles Catalanes, 97-117, 2ª pl. of. 18
08940 - Cornellà de Llobregat (Barcelona)

Tel: 934 246 401
Fax: 934 231 576

e-mail: info@ediciones-eni.com
http://www.ediciones-eni.com

Autor: Michèle AMELOT
Edición española: Claudio SANCHEZ
con la colaboración de Guillermo SOM ("El GUILLE")
Colección **Recursos informáticos** dirigida por Joëlle MUSSET

Copyrighted material

Podrá descargar algunos elementos de este manual en la página web de Ediciones ENI: **http://www.ediciones-eni.com**. Haga clic en el signo ⊞ del cuadro **Nuestras colecciones**. Haga clic en la colección **Recursos Informáticos**, y, a continuación en **VBA Excel 2007 - Programar en Excel: Macros y lenguaje VBA**.
En el recuadro **Más información**, haga clic en **Descargar los ejemplos del libro**.

Presentación — Capítulo 1

A. Presentación del lenguaje VBA 9

B. Las macros de Excel 11

C. Asignación de macros 21

D. El entorno de desarrollo VBE 25

E. Configuración del Editor VBA 30

F. Novedades de la versión 2007 33

El lenguaje VBA — Capítulo 2

A. Módulos 38

B. Procedimientos 40

C. Variables 51

D. Estructuras de decisión 66

E. Estructuras en ciclo 72

Copyrighted Material

F. Operadores 78

G. Reglas de escritura del código 82

La programación de objetos en Excel — Capítulo 3

A. Presentación 86

B. El modelo de objetos de Excel 87

C. Principios de uso de los objetos y las colecciones . . . 91

D. Instrucciones usadas con los objetos 99

E. El Examinador de objetos 102

Objetos de Excel — Capítulo 4

A. El objeto Application 109

B. Objeto Workbook 130

C. El objeto Worksheet 146

D. El objeto Range 156

E. Ejemplos de uso de los objetos 174

Copyrighted Material

Cuadros de diálogo Capítulo 5

A. Presentación 180

B. Cuadros de diálogo integrados 180

C. Cuadros de diálogo predefinidos 184

Formularios Capítulo 6

A. Presentación 194

B. Crear un formulario 194

C. Personalizar un formulario 204

D. Ejemplo de formulario personalizado 211

Mejoras en la interfaz de usuario Capítulo 7

A. Presentación 219

B. Ejemplos de barras de comandos 220

C. Barras de comandos 221

D. Controles (opciones o botones de comandos) de las barras de comandos 224

E. Ejemplos de menús personalizados 230

Copyrighted material

Administración de eventos — Capítulo 8

A. Presentación 242

B. Escritura de eventos 242

C. Eventos del objeto Application 250

D. Eventos del objeto Workbook 253

E. Eventos del objeto Worksheet 256

F. Eventos del objeto Chart 257

Depuración y administración de errores — Capítulo 9

A. Diferentes tipos de error 260

B. Depuración 263

C. Administración de errores en VBA 265

Vínculos entre aplicaciones — Capítulo 10

A. La tecnología Automation 273

B. Comunicación con Word desde Excel 276

C. Comunicación con Access desde Excel 283

Copyrighted material

D. Comunicación con Outlook desde Excel 288

E. El protocolo DDE 290

F. Objetos vinculados o incrustados 294

G. Métodos y propiedades relativos a los vínculos con Excel 296

Internet — Capítulo 11

A. Consultas por Internet 303

B. El objeto QueryTable 305

C. Publicación de páginas Web 312

D. Los objetos WebOptions y DefaultWebOptions . . . 315

E. Importación, exportación y asignación de archivos XML 319

F. El objeto HyperLink 323

Programación Windows — Capítulo 12

A. Presentación de las API 326

B. Llamar a una función de la API Windows 327

Copyrighted material

C. Lista de funciones API Windows 328

D. Ejemplos de uso de funciones API Windows 329

E. El objeto FileSystemObject 332

Código de una miniaplicación Capítulo 13

A. Presentación general 336

B. Descripción de la aplicación GestPresupuesto 337

C. Módulo ThisWorbook 338

D. Formulario NuevoPresupuesto 341

E. Formulario BuscarPresupuesto 346

F. Módulos ProcActions y ProcGene 350

Anexos

A. Lista de instrucciones 357

B. Lista de funciones 365

C. Constantes VBA 383

Índice 387

Copyrighted material

Capítulo 1: Presentación

A. Presentación del lenguaje VBA 9

1. Objetivos del lenguaje VBA 9
2. Algunas definiciones 10
3. Escritura de código VBA 10

B. Las macros de Excel 11

1. Mostrar la ficha Programador en la cinta de opciones . . . 11
2. Descripción de la ficha Programador 11
   a. Grupo "Código" 11
   b. Grupo "Controles" 12
3. Grabación de macros 12
   a. Grabar la primera macro 12
   b. Ejecutar una macro 13
   c. Grabar una macro con referencias relativas 14
   d. Definir el lugar de almacenamiento de una nueva macro 15
   e. Eliminar una macro 16
   f. Grabar un libro con macros 16
4. Las macros y la seguridad 18
   a. Modificación de la configuración de seguridad 18
   b. Descripción de las diferentes opciones de seguridad . . 18
   c. Firmas electrónicas de macros 19

Copyrighted material

5. Modificar el código de una macro 20

C. Asignación de macros 21

1. Asociar una macro a un botón de comando 21
2. Asignar una macro a una imagen 22
3. Asociar una macro a una zona de un objeto gráfico 22
4. Asociar una macro a una imagen Control ActiveX 23
5. Asociar una macro a un icono de la barra de herramienta de acceso rápido 24

D. El entorno de desarrollo VBE 25

1. Acceso al entorno VBE 25
2. Cerrar el entorno VBE 26
3. Volver a Excel 26
4. Descripción del entorno VBE 27

E. Configuración del Editor VBA 30

1. Configuración de la tipografía 30
2. Configuración de la introducción de código 30
3. Gestión de errores 31
4. Acople de una ventana 32
5. Elegir las ventanas que hay que mostrar 33

F. Novedades de la versión 2007 33

1. Novedades de Excel 2007 33
2. Las novedades de VBA Excel 2007 34

Copyrighted material

A. Presentación del lenguaje VBA

Visual Basic para Aplicaciones (VBA) es el **lenguaje de programación** común a todas las aplicaciones del paquete Microsoft Office 2007 (Word, Access, Excel, Outlook y PowerPoint).

1. Objetivos del lenguaje VBA

Trabajando en Excel, el uso del lenguaje VBA le permite:

- **automatizar acciones repetitivas**: VBA permite efectuar en una única operación una cantidad importante de comandos de Excel.
- **interactuar sobre los libros de Excel**: el contenido y la presentación de todos los elementos contenidos en un libro (hojas, celdas, gráficos, etc.) pueden modificarse a través de código VBA.
- **crear formularios personalizados**: los formularios son los cuadros de diálogo compuestos por controles ActiveX (cuadros de texto, listas desplegables, etc.) a los que se les puede asociar código VBA. Los formularios permiten crear interfaces amigables para la entrada o la salida de información.
- **personalizar la interfaz de Excel**: usted puede agregar a la interfaz de Excel nuevos menús y comandos. El código VBA puede asociarse a métodos abreviados de teclado, iconos, etc.
- **modificar las opciones de Excel**: a cada opción de Excel le corresponde una propiedad de un objeto VBA. Por ejemplo, puede modificar el tipo de fuente por defecto a partir de las propiedades StandardFont (fuente) y StandardFontsize (tamaño de fuente) del objeto Application.

Ejemplo:

Application.StandardFont = "Arial"
Application.StandardFontSize = "10"

- **comunicar Excel con otras aplicaciones Microsoft Office**: VBA permite intercambiar información entre las aplicaciones Office usando objetos específicos propios de cada uno. Por ejemplo, puede insertar una tabla o un gráfico Excel en un archivo de Word, crear mensajes Outlook con un archivo Excel adjunto, etc.

© Editions ENI - All rights reserved

Copyrighted material

2. Algunas definiciones

Proyecto

Cada libro abierto en Excel tiene asociado un proyecto que contiene todos los módulos de código VBA agrupados en categorías.

Módulo

Los módulos contienen las macros grabadas y sus propios procedimientos y funciones escritos en VBA. Los módulos pueden exportarse como archivos independientes para luego ser importados en otros libros.

Procedimiento

Los procedimientos son subprogramas escritos en VBA. Cada macro grabada genera un procedimiento con el mismo nombre de la macro. De la misma manera, usted puede crear procedimientos usando la instrucción **Sub**.

Función

Las funciones son procedimientos que devuelven un valor. Para crear una función se debe utilizar la instrucción **Function**.

3. Escritura de código VBA

Hay dos maneras de crear un procedimiento VBA:

- generar automáticamente el código a partir de **la grabación de macros**,
- escribir directamente el código del procedimiento en el **Editor de Visual Basic** (o entorno VBE).

La primera solución es más sencilla pero mucho más limitada que la segunda. Los procedimientos generados automáticamente sólo permiten automatizar acciones repetitivas realizadas con Excel (formato de celdas, ordenar datos, etc.).

Si desea efectuar operaciones específicas: algoritmos de cálculo, intercambio de mensajes y de información con el usuario, controlar la coherencia de datos en un libro o cualquier otra operación que haga uso de estructuras repetitivas o condicionales, usted debe crear sus propios procedimientos en el editor de VBA.

Copyrighted Material

B. Las macros de Excel

1. Mostrar la ficha Programador en la cinta de opciones

Para escribir macros, ejecutar macros grabadas o crear aplicaciones Excel, debe mostrar la ficha **Programador** de la siguiente manera:

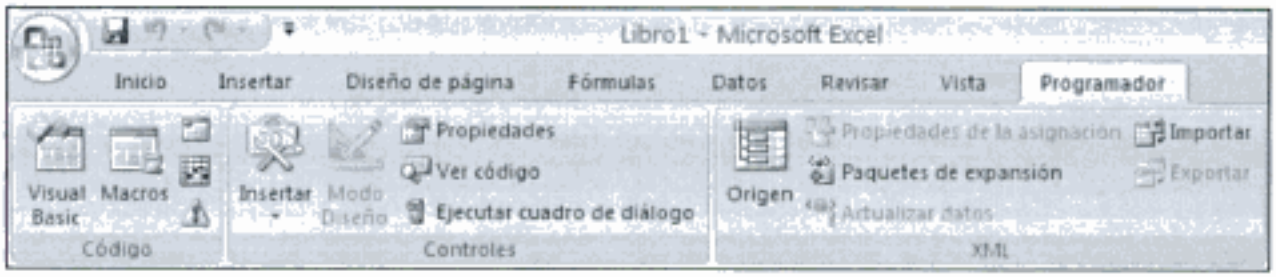
→ Haga clic en el botón **Microsoft Office**, luego en el botón Opciones de Excel (ubicado bajo el cuadro de diálogo).

→ Seleccione la categoría **Más frecuentes** en el menú de la izquierda.

→ Active la opción **Mostrar ficha Programador en la cinta de opciones** ubicada en **Opciones principales para trabajar con Excel**.

– Haga clic en el botón **Aceptar**: la ficha **Programador** se añadirá a la cinta de opciones de Excel, a la derecha de la ficha **Vista**.

2. Descripción de la ficha Programador



a. Grupo "Código"

Nombre del botón	Descripción
Visual Basic	Abre el entorno de desarrollo. El método abreviado de teclado es [Alt] [F11].
Macros	Muestra la lista de macros. El método abreviado de teclado es [Alt] [F8].
Grabar macro	Comienza la grabación de una macro.

© Editions ENI - All rights reserved

Copyrighted material

Usar referencias relativas	Permite usar referencias relativas a la primera celda seleccionada.
Seguridad de macros	Personaliza la configuración de seguridad de las macros.

b. Grupo "Controles"

Nombre del botón	Descripción
Insertar	Permite insertar controles (formularios o ActiveX) en Excel.
Modo Diseño	Activa o desactiva el modo Diseño. En el modo Diseño los controles ActiveX pueden seleccionarse y modificarse pero no pueden ejecutarse.
Propiedades	Muestra las propiedades del objeto Excel seleccionado (hoja o control).
Ver código	Permite acceder directamente al código asociado al control seleccionado.
Ejecutar cuadro de diálogo	Ejecuta un cuadro de diálogo personalizado.

El grupo XML permite administrar asignaciones XML.

3. Grabación de macros

a. Grabar la primera macro

Para crear una macro con el grabador:

- Haga clic en el botón de la ficha **Programador** o en el botón de la barra de estado.
- En el cuadro de diálogo que aparece, escriba el nombre de la macro, su descripción y, si lo desea, el método abreviado de teclas asociado.

Copyrighted material

- Haga clic en el botón **Aceptar** para iniciar la grabación.
- Realice en Excel las operaciones que desee grabar. Por ejemplo, seleccione el rango de celdas "A1:B8", y aplique un relleno y un borde a esas celdas.
- Haga clic en el botón Detener grabación de la ficha **Programador** o en el botón de la barra de estado para detener la grabación de la macro.

b. Ejecutar una macro

Para ejecutar una macro desde Excel:

- Haga clic en el botón Macros de la ficha **Programador** o pulse el método abreviado de teclas [Alt] [F8].
- Haga doble clic en el nombre de la macro o primero selecciónela y luego haga clic en el botón **Ejecutar**.

Para visualizar la información relativa a una macro (método abreviado de teclado y descripción), seleccione la macro en la lista y haga clic en el botón **Opciones**.

© Editions ENI - All rights reserved

Copyrighted material

Para detener la ejecución de una macro, pulse [Escape] o [Ctrl][Pausa].

c. Grabar una macro con referencias relativas

Si graba una macro en el modo **referencias absolutas** (modo por defecto), los rangos de celdas referenciados en las operaciones de selección, desplazamientos... serán fijos. Por ejemplo: Range("A2") designa la celda A2.

Si graba una macro en el modo **referencias relativas**, los rangos de celdas serán expresados en relación a la posición de la primera celda activa.
Por ejemplo: ActiveCell.range("A2") designa la celda ubicada bajo la celda activa, ActiveCell.range("B1") designa la celda ubicada a la derecha de la celda activa.

ActiveCell.range("A1") siempre hace referencia a la primera celda activa. A1 puede considerarse la referencia relativa a la primera celda activa.

Para grabar una macro con referencias relativas:

- Haga clic en el botón de la ficha **Programador**: el botón quedará activo (se verá de color naranja).
- Si vuelve a hacer clic en el botón , éste quedará desactivado y las macros se grabarán con referencias absolutas.

Ejemplo

*La misma secuencia de operaciones se ha registrado en dos macros: la primera (RefRelativa) se grabó con la opción **referencias relativas**, la segunda (RefAbsoluta), con la opción **referencias absolutas**.*

Copyrighted material

La secuencia de operaciones es la siguiente:

– selección de un rango de celdas,

– desplazar el rango dos filas hacia abajo y una columna a la derecha.

```
Sub RefAbsoluta()
'  Referencias absolutas
   Range("B2:C8").Select
   Selection.Cut Destination:=Range("C4:D10")
   Range("C4:D10").Select
End Sub
```

```
Sub RefRelativa()
'  Referencias relativas
   ActiveCell.Range("A1:B7").Select
   Selection.Cut Destination:=ActiveCell.Offset(2, 1).Range("A1:B7")
   ActiveCell.Offset(2, 1).Range("A1:B7").Select
End Sub
```

d. Definir el lugar de almacenamiento de una nueva macro

Para definir el lugar de almacenamiento de una nueva macro:

→ Haga clic en el botón

de la ficha **Programador** o pulse el método abreviado [Alt] [F8].

→ Abra la lista **Grabar macro en** y seleccione el libro en el que desee crear la macro.

Si elige **Libro de macros personal**, la macro se grabará en el libro personal.xlsb; la macro será accesible desde todos los libros Excel.

© Editions ENI - All rights reserved

Copyrighted Material

e. Eliminar una macro

Para eliminar una macro:

- Haga clic en el botón Macros de la ficha **Programador** o pulse el método abreviado [Alt] [F8].
- Seleccione la macro que desea eliminar y haga clic en el botón **Eliminar**. Haga clic en el botón **Sí** para confirmar la eliminación.

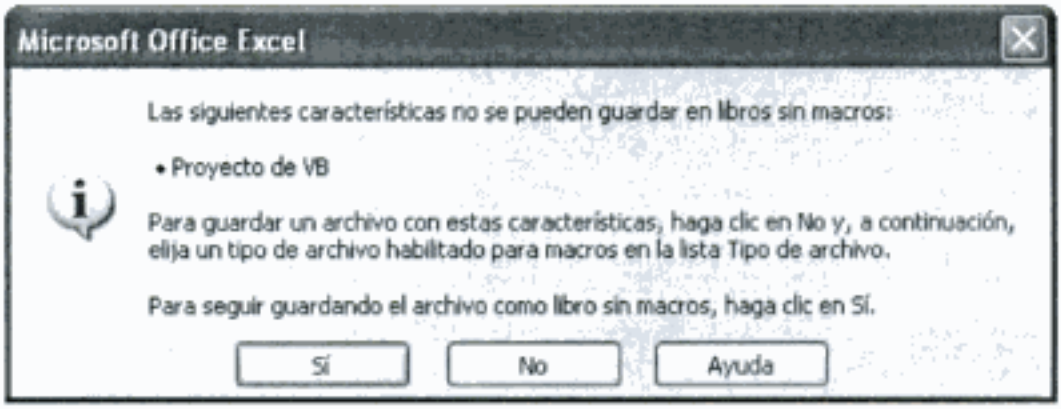
f. Grabar un libro con macros

Para grabar un libro con macros:

- Haga clic en el botón Macros de la ficha **Programador** o pulse el método abreviado [Alt] [F8].
- Haga doble clic en el nombre de la macro o pulse la combinación de teclas asociada a la macro.

Para visualizar la información relativa a una macro (método abreviado de teclado y descripción), seleccione la macro de la lista y haga clic en el botón **Opciones...**.

Si guarda por primera vez un libro con macros, aparecerá el mensaje siguiente:

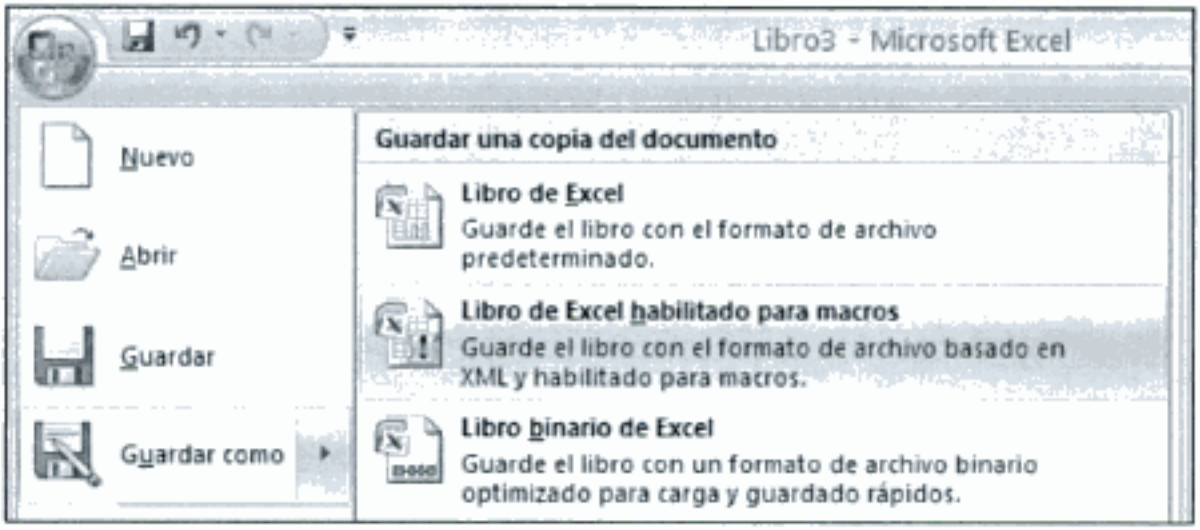


- Haga clic en el botón **No** para anular la grabación del libro sin las macros.
- En el cuadro de diálogo **Guardar como**, abra la lista **Tipo de archivo** y seleccione **Libro de Excel habilitado para macros (*.xlsm)**.

Los libros que contienen macros tienen la extensión xlsm (en vez de xslx) y su icono se distingue por un signo de exclamación.

Para grabar un nuevo libro con macros, también puede realizar las siguientes operaciones:

- Haga clic en el botón **Microsoft Office** .
- Arrastre el puntero del ratón hasta la opción **Guardar como**.
- Haga clic en **Libro de Excel habilitado para macros**.



© Editions ENI - All rights reserved

Copyrighted material

4. Las macros y la seguridad

La configuración de seguridad de las macros permite controlar lo que ocurre al abrir un libro que contiene macros.

Las modificaciones de la configuración de seguridad de macros rigen solamente en Excel y no afectan a las otras aplicaciones de Microsoft Office.

a. Modificación de la configuración de seguridad

- Haga clic en el botón de la ficha **Programador**.
- Seleccione la opción deseada.
- Haga clic en **Aceptar** para validar su elección.

Si se cambia la configuración de seguridad, las nuevas opciones se aplicarán a todos los libros, excepto a los libros ya abiertos en Excel. Para aplicarlas a estos libros, debe cerrarlos y abrirlos nuevamente.

b. Descripción de las diferentes opciones de seguridad

Deshabilitar todas las macros sin notificación
Todas las macros y todas las advertencias de seguridad serán deshabilitadas.

Deshabilitar todas las macros con notificación
Opción por defecto. Las macros serán deshabilitadas pero aparecerá la siguiente advertencia de seguridad (bajo la cinta de opciones) para los libros que contienen macros:

También es posible habilitar las macros, solamente para este libro, de la siguiente manera:

- Haga clic en el botón Opciones... ubicado a la derecha de la advertencia de seguridad anterior.

Copyrighted material

→ Seleccione la opción **Habilitar este contenido** y haga clic en el botón **Aceptar**: las macros serán habilitadas y desaparecerá la advertencia de seguridad.

Deshabilitar todas las macros excepto las firmadas digitalmente
Si las macros de un libro tienen firma digital de un origen aprobado, las macros podrán ejecutarse. Si el origen no está autorizado, aparecerá una notificación: en este caso es posible habilitar las macros firmadas o aprobar el origen. Las macros sin firma digital no podrán habilitarse.

Habilitar todas las macros (no recomendado; puede ejecutarse código posiblemente peligroso).
Si se selecciona esta opción, todas las macros serán habilitadas. Se recomienda no usar esta opción de forma permanente.

Cualquiera que sea la opción elegida, si se instala un programa antivirus compatible con Microsoft Office System 2007, los libros que contengan macros serán analizados antes de abrirse.

---

c. Firmas electrónicas de macros

Para firmar digitalmente un proyecto macro, debe:

- Obtener un certificado digital de parte de una autoridad de certificación comercial, como Verisign (www.verisign.com), e instalarlo,
- Firmar digitalmente su proyecto. Para ello, pase al entorno VBE (accesible desde el comando

de la ficha **Programador**) y seleccione la opción **Firma digital** del menú **Herramientas**. Haga clic en el botón **Elegir** para seleccionar su certificado.

Para probar sus proyectos macros, en su ordenador, puede crear su propio certificado de autofirma con la ayuda de Selfcert.exe (programa distribuido con Microsoft Office 2007).

© Editions ENI - All rights reserved

Copyrighted material

5. Modificar el código de una macro

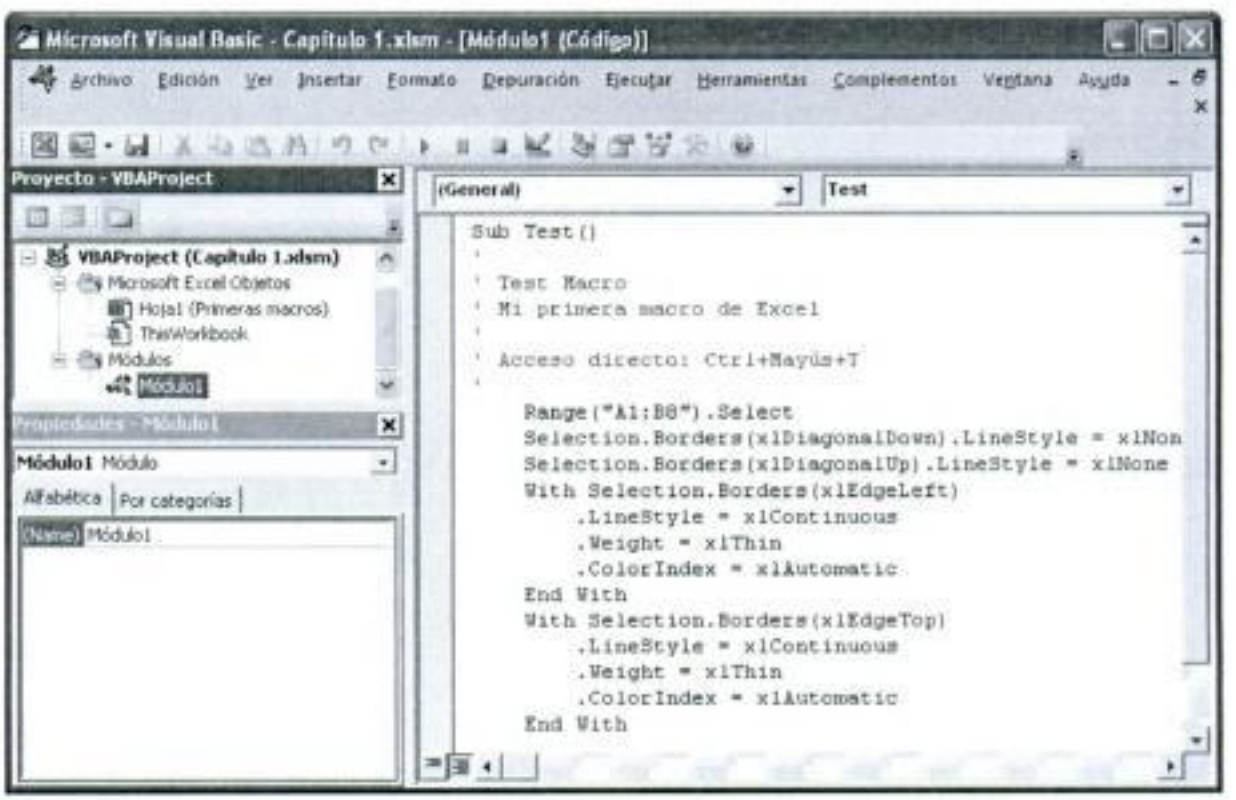
Para acceder al código de una macro:

- Haga clic en el botón

de la ficha **Programador** o pulse el método abreviado [Alt] [F8].
- Seleccione la macro que desea modificar y haga clic en el botón **Modificar**.

El código de la macro aparecerá en una ventana del entorno Microsoft Visual Basic (ver. D. El entorno de desarrollo VBE).

Una macro corresponde a un **procedimiento** en lenguaje VBA. Un procedimiento VBA es una secuencia de instrucciones agrupadas en un bloque de código que comienza con **Sub** y termina con **End Sub**.



Puede modificar o completar el código dentro de este procedimiento.

Copyrighted Material

Ejemplo

Inserte el siguiente código al final del procedimiento:

```
...
    MsgBox "El formato de celdas " & Selection.Address _
            & vbCr & "se ha modificado correctamente"
End Sub
```

Para probar su procedimiento, haga clic en el icono o use la tecla [F5]. Aparecerá el siguiente mensaje:

C. Asignación de macros

Se puede asociar una macro a distintos controles (botones de comando, imágenes, etc.) o a la barra de herramientas de acceso rápido para dar al usuario un acceso amigable y rápido para ejecutar macros.

1. Asociar una macro a un botón de comando

- Haga clic en el botón

de la ficha **Programador**.

- Dibuje el botón con la ayuda de la herramienta (parte superior izquierda de la barra de Controles de formulario).

  Al soltar el botón del ratón, aparecerá el cuadro de diálogo **Asignar macro**.

© Editions ENI - All rights reserved

Copyrighted material

- Seleccione el **Nombre de la macro** en la lista y haga clic en **Aceptar**.

2. Asignar una macro a una imagen

- Inserte una imagen con un clic en el botón Imagen de la ficha **Insertar**.
- Haga clic con el botón secundario del ratón y seleccione la opción **Asignar macro**.
- Seleccione el **Nombre de la macro** en la lista y haga clic en el botón **Aceptar**.

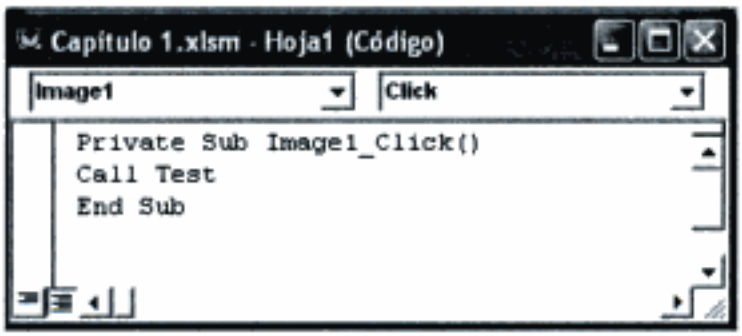
3. Asociar una macro a una zona de un objeto gráfico

- Inserte un objeto gráfico (imagen, imagen prediseñada, forma o SmartArt, accesibles desde el grupo **Ilustraciones** de la ficha **Insertar**).
- Para definir en el objeto una zona que permita ejecutar una macro, haga clic en Formas (en la ficha **Insertar**), seleccione la forma deseada y dibuje la forma dentro del objeto que acaba de crear.
- Haga clic con el botón derecho del ratón y seleccione la opción **Asignar macro**.
- Seleccione el **Nombre de la macro** en la lista y valide con un clic en el botón **Aceptar**.
- Seleccione la forma insertada. Aparecerá la sección **Herramientas de dibujo** y haga clic en la ficha **Formato**.
- En el grupo **Estilos de forma** de la ficha **Formato**, haga clic en la lista Relleno de forma y luego en Sin relleno.
- En el mismo grupo **Estilo de forma** de la ficha **Formato**, haga clic en la lista Contorno de forma, luego en Sin contorno.
- Ahora, al hacer clic en la forma, se ejecutará la macro.

Copyrighted Material

4. Asociar una macro a una imagen Control ActiveX

- Active el modo Diseño (si es que no está ya activo) con un clic en el botón Modo Diseño de la ficha **Programador**.
- Haga clic en el botón Insertar de la ficha **Programador**.
- Dibuje una imagen con la ayuda de la herramienta (dentro de los controles ActiveX) y suelte el botón del ratón.
- Ahora puede mostrar y modificar las propiedades de la imagen (en particular, asociar una imagen por medio de la propiedad **Picture**) con un clic en el botón Propiedades.
- Haga clic con el botón derecho del ratón y seleccione la opción **Ver código**.
- Dentro del procedimiento VBA asociado a la imagen, escriba la instrucción **Call** seguida del nombre de la macro que desea ejecutar.



- Desactive el modo Diseño con un clic en el botón Modo Diseño de la ficha **Programador**.
- Haga clic en la imagen: la macro Test se ejecuta.

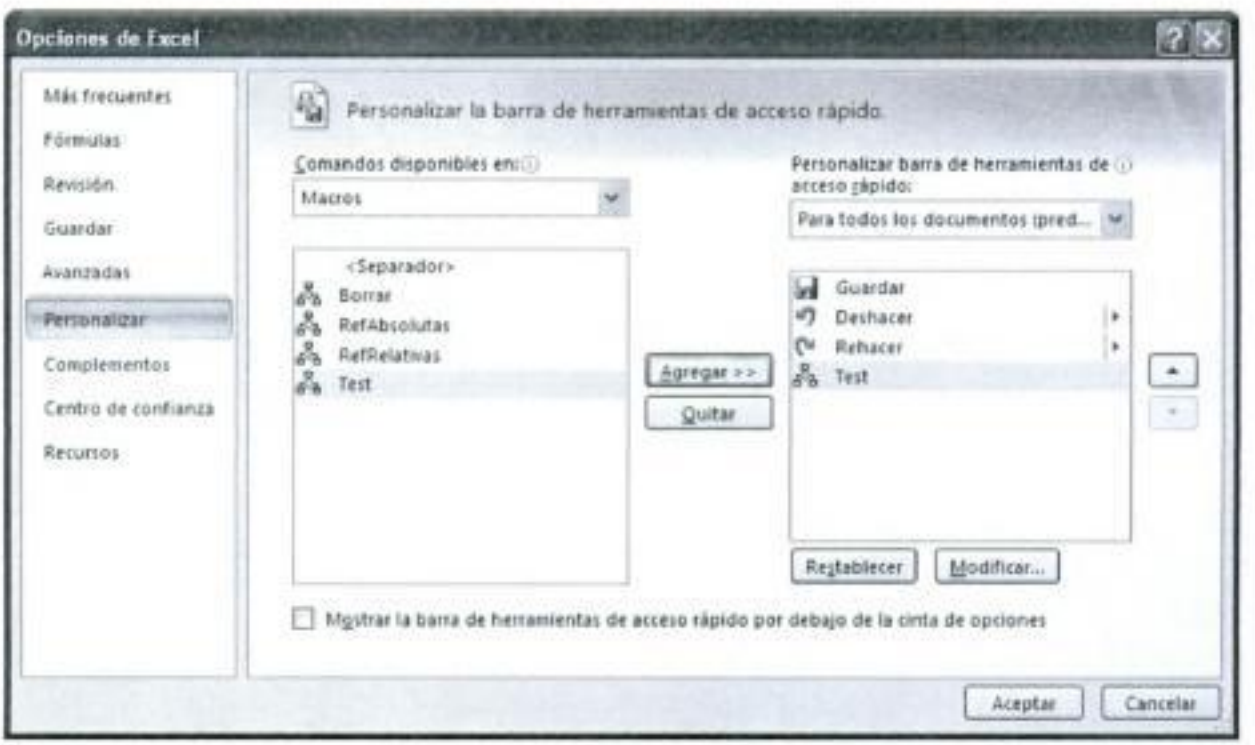
© Editions ENI - All rights reserved

Copyrighted material

5. Asociar una macro a un icono de la barra de herramienta de acceso rápido

También es posible insertar en la barra de herramienta de **acceso rápido** un comando que permita ejecutar una macro. Para ello:

- Haga clic en el botón **Microsoft Office** , luego en el botón **Opciones de Excel** (abajo y a la derecha del cuadro de diálogo).
- Seleccione la categoría **Personalizar** en el menú de la izquierda.
- En la lista desplegable de categorías, arriba y a la izquierda, seleccione **Macros**: aparecerán las macros disponibles en la columna de la izquierda.
- Seleccione una macro y haga clic en el botón **Agregar**: la macro pasará a la columna de la derecha.



Copyrighted material

- Haga clic en el botón **Modificar** para seleccionar un nuevo icono para la macro.

  Puede modificar el orden de los comandos por medio de las dos flechas situadas a la derecha.

- En la lista desplegable situada arriba a la derecha, puede seleccionar su libro: en este caso las modificaciones efectuadas solamente se aplicarán a ese libro.

- Haga clic en el botón **Aceptar** para validar.

  Ahora habrá un nuevo comando en la barra de herramienta de **acceso rápido** para ejecutar la macro.

D. El entorno de desarrollo VBE

VBE (*Visual Basic Editor*) es el entorno en el que puede introducir, modificar y probar su código VBA. Este entorno se llama también IDE (*Integrated Development Environment*) o editor de VBA.

El entorno VBE pone a su disposición numerosas herramientas para facilitar la programación y la puesta a punto de su código VBA: herramientas de depuración, introducción asistida, explorador de objetos, etc.

1. Acceso al entorno VBE

- Para acceder al entorno VBE desde Excel, haga clic en el botón

de la ficha **Programador** o pulse la combinación [Alt] [F11].

© Editions ENI - All rights reserved

Copyrighted material

Si el entorno VBE ya está abierto, puede acceder a ello desde la barra de tareas Windows:

2. Cerrar el entorno VBE

Para cerrar el entorno VBE y volver a Excel, haga clic en el aspa roja ubicada arriba a la derecha de la ventana principal del entorno VBE.

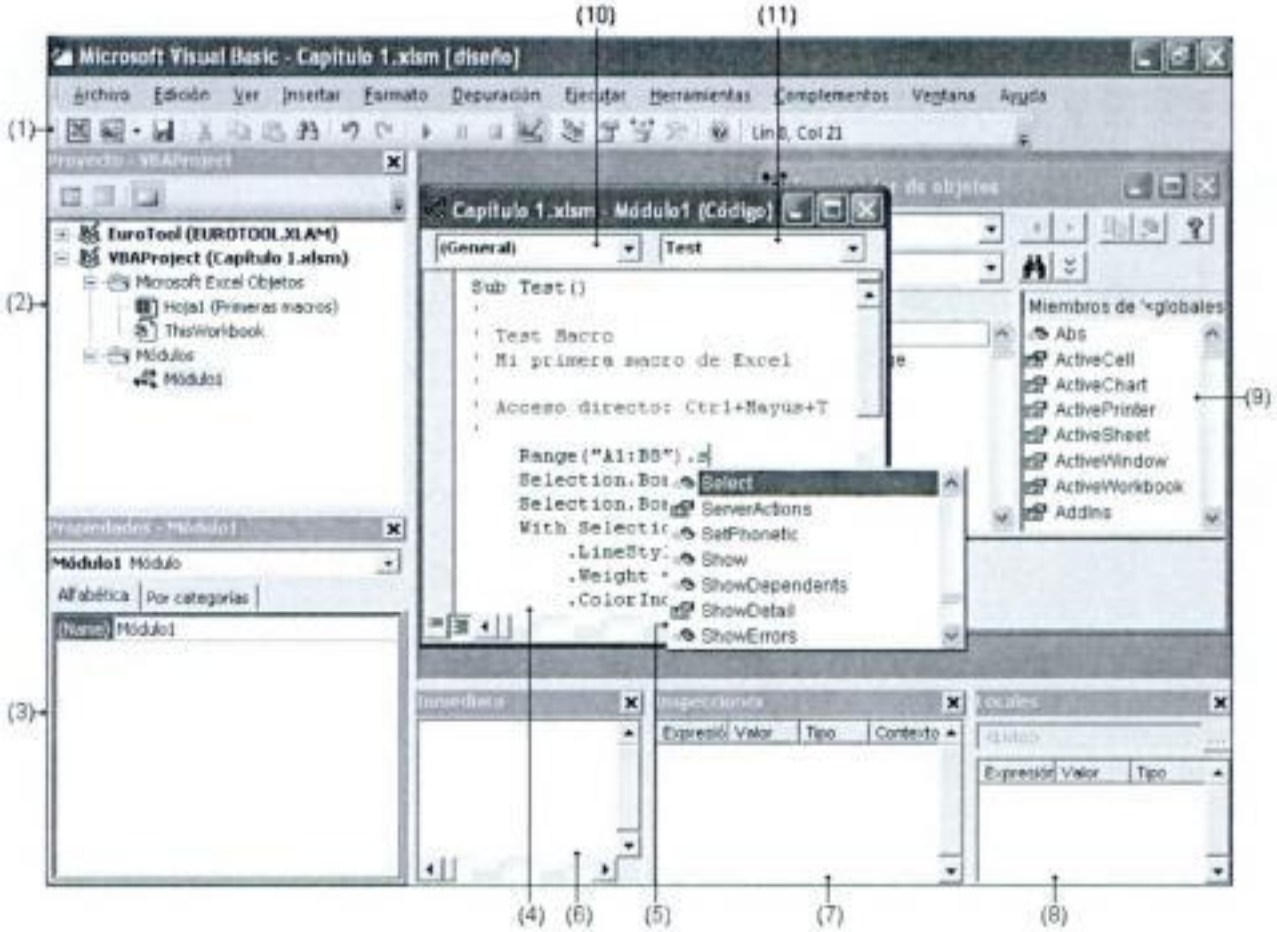
3. Volver a Excel

Para volver a Excel sin cerrar el entorno VBE, use cualquiera de estas tres posibilidades:

- Haga clic en el icono de la barra de herramientas **Estándar**,
- O pulse el método abreviado [Alt] [F11].

Copyrighted material

4. Descripción del entorno VBE



Todas las ventanas del entorno VBE pueden visualizarse desde el menú **Ver**.

© Editions ENI - All rights reserved

Copyrighted Material

(1) La barra de herramientas Estándar

1 Ver Microsoft Excel
2 Insertar una hoja
3 Guardar libro
4 Cortar
5 Copiar
6 Pegar
7 Buscar
8 Deshacer
9 Repetir
10 Ejecutar Sub
11 Interrumpir
12 Restablecer
13 Modo de diseño
14 Explorador de proyectos
15 Ventana de Propiedades
16 Examinador de objetos
17 Cuadro de herramientas
18 Ayuda de Microsoft Visual Basic
19 Posición actual en la ventana de código

(2) El explorador de proyectos

Cada libro abierto en Excel tiene asociado un proyecto. El explorador de proyectos permite ver todos los proyectos y todos los módulos de cada proyecto según una estructura en árbol. Los módulos se agrupan en cuatro categorías:

- módulos asociados a objetos Excel (libro y hojas),
- módulos asociados a formularios,
- módulos estándares,
- módulos de clase.

Cada módulo puede contener muchos procedimientos.

(3) La ventana Propiedades

Permite mostrar las propiedades relativas al libro, a las hojas de cálculo, a las hojas gráficas y a los formularios.

Copyrighted material

(4) La ventana Código

En esta ventana se encuentran dos zonas con listas desplegables:

- la zona objeto (10) muestra la lista de los objetos de módulo,
- la zona procedimiento (11) muestra los procedimientos o los eventos del objeto seleccionado en la zona objeto. Los eventos ya usados aparecen en negrita.

(5) Instrucciones que se completan automáticamente

Una lista desplegable aparece automáticamente cuando se escribe el nombre de un objeto seguido de un punto. La lista muestra los métodos, propiedades y constantes disponibles para ese objeto.

Si la lista no está activa, seleccione **Opciones** en el menú **Herramientas** y marque la casilla **Lista de miembros automática** de la ficha **Editor**.

(6) Ventana Inmediato

Permite mostrar los valores de las variables, modificarlas, y ejecutar las instrucciones.

(7) Ventana Locales

Contiene todos los valores de las variables del procedimiento en curso.

(8) Ventana Inspección

Contiene todos los valores de las variables que previamente hayan sido definidas como expresiones de inspección.

Las ventanas **Inmediato**, **Locales** e **Inspección** se usan principalmente para la depuración de aplicaciones (ver capítulo 9).

(9) El Explorador de objetos

Permite visualizar, para cada objeto, sus propiedades, métodos y constantes.

© Editions ENI - All rights reserved

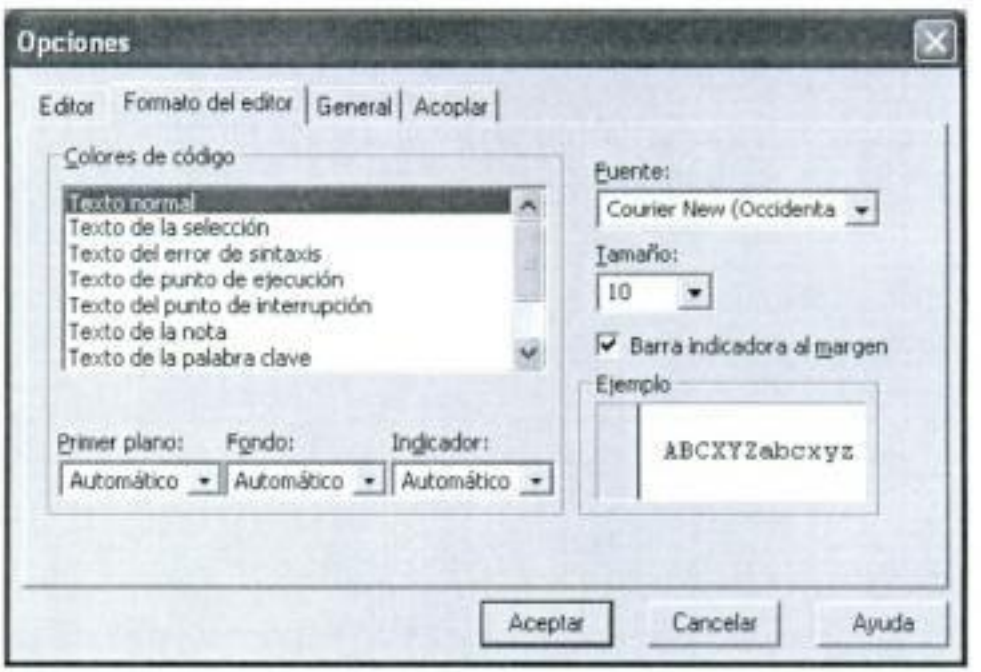
Copyrighted material

E. Configuración del Editor VBA

1. Configuración de la tipografía

Las palabras clave, las funciones y las instrucciones VBA aparecen en azul; los objetos, métodos y propiedades, en negro, y los comentarios, en verde. Las instrucciones que contienen errores se destacan en rojo.

→ Para modificar el estilo (color, tipo de letra, tamaño) de los diferentes tipos de código, seleccione **Opciones** en el menú **Herramientas** y haga clic en la ficha **Formato del editor**.

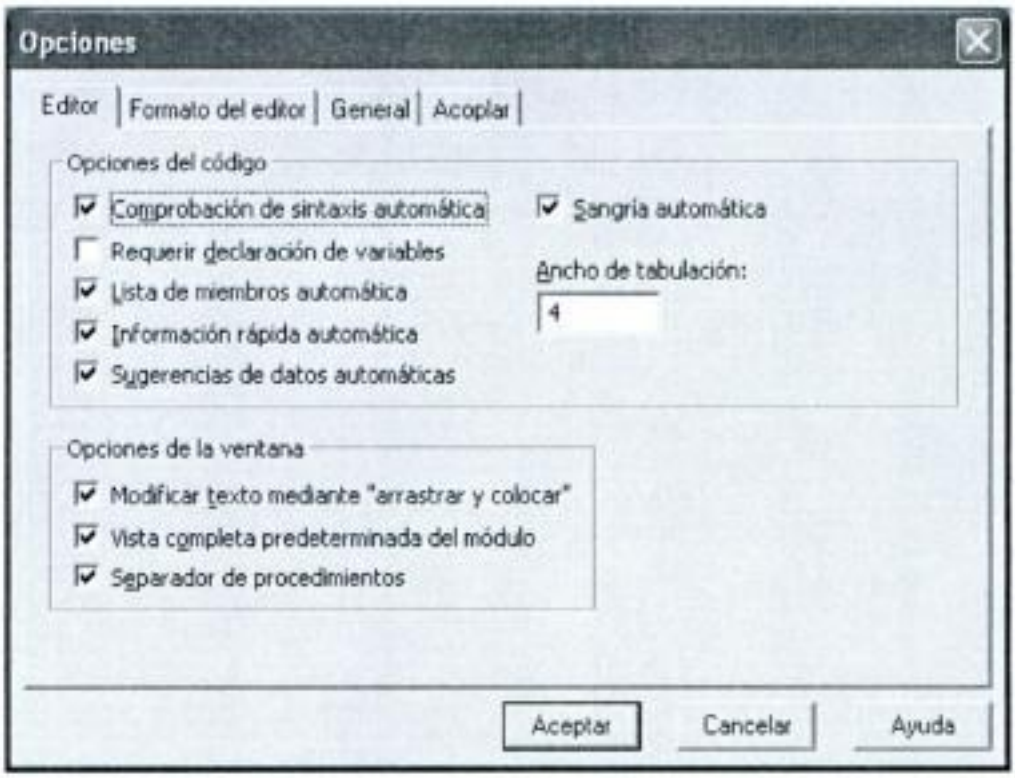


2. Configuración de la introducción de código

Existen diferentes herramientas que facilitan la introducción y la actualización del código VBA: por ejemplo, la comprobación automática de la sintaxis, la declaración obligatoria de las variables, las instrucciones que se completan automáticamente, etc.

→ Para activar estas opciones, seleccione **Opciones** en el menú **Herramientas** y haga clic en la ficha **Editor**.

Copyrighted Material



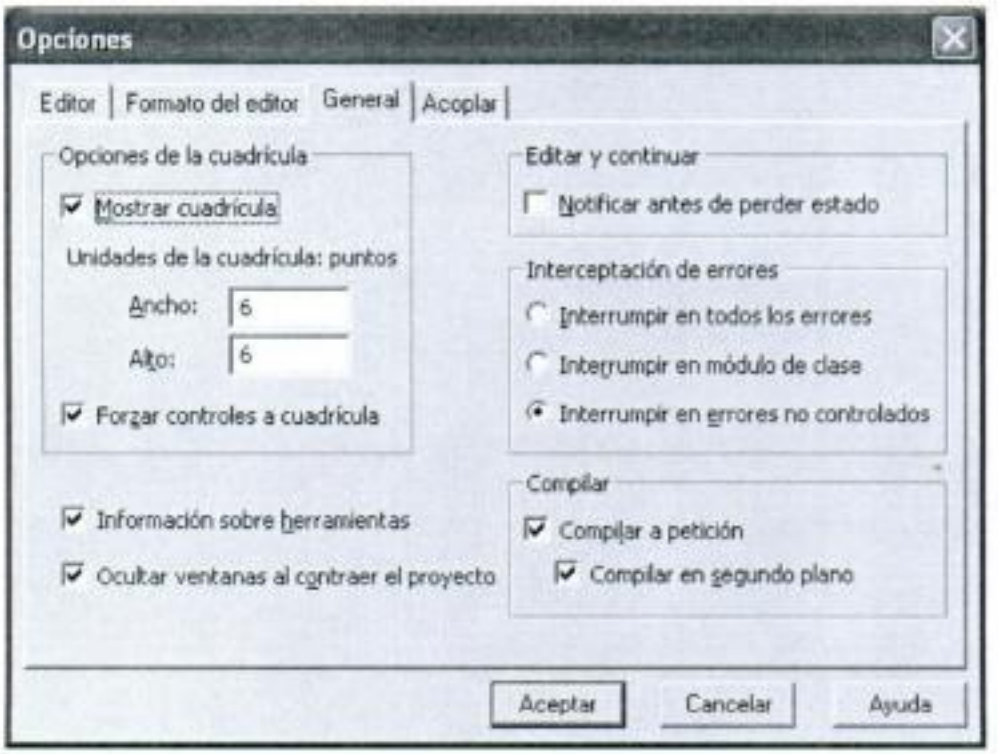
3. Gestión de errores

Las opciones de interceptación de errores permiten establecer si la ejecución del código se interrumpirá al ocurrir un error de ejecución.

→ Para activar esta opción, seleccione **Opciones** en el menú **Herramientas** y haga clic en la ficha **General**.

© Editions ENI - All rights reserved

Copyrighted material



→ Si su código VBA incluye una gestión de errores, seleccione la opción **Interrumpir en errores no controlados** o las instrucciones de gestión de errores no serán tenidas en cuenta.

Ciertos ejemplos de este libro incluyen una gestión de errores; es importante que active esta opción para que funcionen correctamente.

La gestión de errores se explica en el capítulo 9.

---

4. Acople de una ventana

Una ventana acoplada se coloca automáticamente cuando la mueve. Una ventana no está acoplada si puede ubicarse en cualquier lugar de la pantalla y permanecer ahí.

→ Para definir las ventanas que desea acoplar, seleccione **Opciones** en el menú **Herramientas** y haga clic en la ficha **Acoplar**.

→ Active las ventanas que desea acoplar y desactive las otras. Luego haga clic en **Aceptar**.

Copyrighted Material

5. Elegir las ventanas que hay que mostrar

Nombre de la ventana que hay que mostrar	Menú	Barra de herramientas	Teclado
Propiedades	**Ver - Ventana Propiedades**		[F4]
Proyectos	**Ver - Explorador de proyectos**		[Ctrl] [R]
Inmediato	**Ver - Ventana Inmediato**		[Ctrl] [G]
Inspección	**Ver - Ventana Inspección**		
Ventana Locales	**Ver - Ventana Locales**		
Explorador de objetos	**Ver - Examinador de objetos**		[F2]
Módulo	**Ver - Código**		[F7]

F. Novedades de la versión 2007

1. Novedades de Excel 2007

Excel 2007 forma parte del paquete **Microsoft Office 2007** (también llamado Office 12) y se beneficia, en consecuencia, de la mayor parte de las mejoras de esta versión: nueva interfaz de usuario (cinta de opciones y fichas que agrupan los comandos y reemplazan los menús y barras de herramientas), comparte las herramientas gráficas (entre Excel, Word y PowerPoint), temas de Office, grabación de archivos en formato pdf y xps, etc.

Otras novedades que incluye Excel 2007: capacidad general aumentada (aumenta la cantidad de filas y columnas), mayor facilidad en la redacción de fórmulas, mejoras en el formato condicional, tablas Excel, filtros y ordenamientos, etc.

© Editions ENI - All rights reserved

Copyrighted material

2. Las novedades de VBA Excel 2007

Al contrario de lo que ocurre con la interfaz de Excel, el entorno de desarrollo conserva sus menús y barras de herramientas.

De todas formas, la documentación en línea se ha mejorado considerablemente: el aspecto de Visual Studio, ejemplos de código de tareas y modelo de objetos enriquecido (información sobre los cambios en relación a las versiones anteriores).

De la misma forma, las mejoras en Excel introducen un enriquecimiento en el modelo de objetos de Excel, como por ejemplo:

- nuevos objetos asociados a los formatos condicionales: Databar (barra para formato condicional de una barra de datos), IconCriterion (representa el criterio con un icono individual dentro de un juego de iconos), etc.
- nuevos métodos del objeto Workbook: ApplyTheme (permite aplicar un tema a un libro), ExportAsFixedFormat (permite publicar un libro en formato PDF o XPS), etc.
- nuevas propiedades asociadas al objeto Application: ShowDevTools (indica si la ficha **Programador** está visible en la cinta de opciones), ShowMenuFloaties (indica si las minibarras de herramientas deben aparecer cuando el usuario hace clic con el botón derecho del ratón), etc.

Copyrighted Material

Capítulo 2: El lenguaje VBA

A. Módulos 38

1. Presentación 38
2. Acceso a los módulos 39
3. Importar y exportar código VBA 40

B. Procedimientos 40

1. Definiciones 40
2. Acceso a los procedimientos 41
3. Procedimientos Sub 42
4. Procedimientos Function 43
5. Declaración de procedimientos 44
6. Alcance de los procedimientos 44
7. Argumentos de los procedimientos 45
8. Argumentos con nombre 46
9. Llamar a un procedimiento 47
10. Llamar a una función VBA en una fórmula Excel 48
11. Ejemplos de procedimientos y funciones 50

Copyrighted material

C. Variables 51

1. Tipos de variables 52
2. Declaración de variables 57
   a. Declaraciones implícitas 57
   b. Declaraciones explícitas 57
   c. Sintaxis de las instrucciones de declaración 58
3. Declaración de los tipos de variables 59
   a. Declaraciones explícitas del tipo 59
   b. Declaraciones implícitas del tipo 59
4. Matrices 61
5. Constantes 63
   a. Constantes personalizadas 63
   b. Constantes integradas 64

D. Estructuras de decisión 66

1. Instrucción IF 67
2. Instrucción Select Case 70

E. Estructuras en ciclo 72

1. Instrucción Do...Loop 72
2. Instrucción While...Wend 74
3. Instrucción For...Next 74
4. Instrucción For Each...Next 76
5. Salir de las estructuras de control 77

Copyrighted Material

F. Operadores 78

1. Operadores aritméticos 78
2. Operadores de comparación 79
3. Operadores lógicos 79
4. Operador de concatenación 80
5. Prioridad de los operadores 80

G. Reglas de escritura del código 82

1. Comentarios 82
2. Carácter de continuación 82
3. Sangrías 82
4. Nombres de procedimientos, variables y constantes 83

Copyrighted Material

A. Módulos

1. Presentación

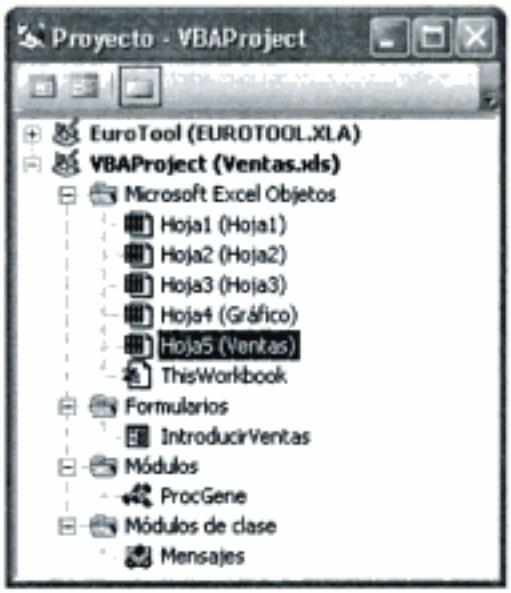
El código VBA asociado a un libro está agrupado en un proyecto que contiene varias carpetas:

La carpeta **Microsoft Excel Objetos**	Contiene un módulo de clase asociado al libro del proyecto (llamado por defecto ThisWorkbook) y un módulo de clase por cada una de las hojas de cálculo u hojas de gráfico del libro. En particular, en estos módulos de clase se encuentran los procedimientos de eventos asociados al libro y a las hojas.
La carpeta **Formularios**	Contiene los formularios (UserForm) del proyecto y el código VBA asociado.
La carpeta **Módulos**	Agrupa los diferentes módulos estándares (compuestos por uno o más procedimientos) que pueden ser llamados desde cualquier procedimiento del proyecto.
La carpeta **Módulos de clase**	Contiene los módulos de clase usados para la creación de nuevas clases de objetos. Los módulos de clase son utilizados especialmente para la escritura de los procedimientos de eventos asociados a los objetos **Application** y **Chart** (ver capítulo 8).

La lista de todos los módulos aparece en forma jerárquica en el Explorador de proyectos del entorno VBE.

Copyrighted material

Si el explorador de proyectos no está a la vista, elija la opción **Explorador de proyectos** del menú **Ver** o pulse el método abreviado [Ctrl] [R].



Para ver el código asociado a un módulo, haga doble clic sobre el nombre del módulo.

Los elementos del lenguaje VBA descritos en este capítulo pueden usarse en los diferentes módulos.

2. Acceso a los módulos

Para insertar un nuevo módulo en el entorno VBE, use la opción **Módulo** del menú **Insertar**, o haga clic en el icono de la barra de herramientas **Estándar** y luego en **Módulo**.

Si la ventana **Módulo** está maximizada, el nombre del módulo aparece sobre la barra de título de **Microsoft Visual Basic**.

→ Para pasar de un módulo a otro, en la ventana **Proyecto**, haga doble clic en el nombre del módulo que desea activar.

© Editions ENI - All rights reserved

Copyrighted material

- Para eliminar un módulo, en la ventana **Proyecto**, haga un clic derecho en el nombre del módulo que desea eliminar, elija la opción **Quitar*modulo*** e indique si desea exportar el módulo o no.

- Para dar nombre a un módulo, active el módulo. Si es necesario, abra la ventana de **Propiedades** y cambie la propiedad **Name**.

3. Importar y exportar código VBA

Los módulos y formularios pueden exportarse a un archivo para luego importarlo a otro proyecto Excel.

- Para exportar un archivo, haga clic sobre el nombre del archivo en el explorador de proyectos, luego seleccione la opción **Exportar archivo** del menú **Archivo** (o del menú contextual) o pulse la combinación de teclas [Ctrl] [E].

- Para importar un archivo, haga clic sobre el nombre del archivo en el explorador de proyectos, luego seleccione la opción **Importar archivo** del menú **Archivo** (o del menú contextual) o pulse la combinación de teclas [Ctrl] [M].

La extensión del archivo creado depende del tipo de archivo exportado:

- Los **módulos de clase** (módulos asociados al libro y a las hojas y módulos de clase independientes) tienen la extensión **.cls**,
- Los **formularios** tienen la extensión **.frm**,
- Los **módulos** estándares tienen la extensión **.bas**.

B. Procedimientos

1. Definiciones

Los procedimientos son **subprogramas** que permiten descomponer una tarea de programación compleja en un conjunto de tareas más breves y simples. Permiten organizar el código dentro de módulos para obtener un código de mantenimiento más simple y fácilmente reutilizable.

Copyrighted material

En VBA Excel, se distinguen tres tipos de procedimientos:

- los procedimientos **Sub** (por subrutina) se llaman subprogramas o procedimientos Sub,
- los procedimientos **Function** se llaman funciones,
- los procedimientos **Property** se llaman procedimientos de propiedad.

En este capítulo, solamente nos interesan los dos primeros, que son los más utilizados.

- Puntos comunes entre procedimientos Sub y funciones:
  - ambos contienen instrucciones y/o métodos VBA,
  - ambos aceptan argumentos,
  - ambos pueden ser llamados desde otras funciones u otros procedimientos Sub.
- Característica específica de las funciones:
  - devuelven valores.

2. Acceso a los procedimientos

→ Para acceder a un procedimiento desde la ventana de código de un módulo, abra la segunda lista de la ventana del módulo, haga clic sobre el nombre del procedimiento al que desea acceder o recorra los procedimientos con [Ctrl] [Flecha arriba] y [Ctrl] [Flecha abajo].

→ Para seleccionar una palabra, haga doble clic sobre la palabra.

→ Para seleccionar una línea, ubique el puntero del ratón a la izquierda de la línea y haga clic cuando el puntero se convierte en una flecha.

→ Para seleccionar un grupo de caracteres, use la técnica de arrastre y soltar o haga [Mayús] clic.

→ Para seleccionar un procedimiento completo, ubique el puntero del ratón a la izquierda de cualquiera línea del procedimiento. Cuando el puntero se convierte en una flecha, haga doble clic.

© Editions ENI - All rights reserved

Copyrighted Material

- Para ejecutar un procedimiento, haga clic en el procedimiento que desea ejecutar y pulse [F5] o .
- Para eliminar un procedimiento, seleccione todo el procedimiento, y pulse [Supr].

3. Procedimientos Sub

Hay dos tipos de procedimientos Sub:

- los procedimientos Sub generales,
- los procedimientos Sub asociados a eventos.

Un **procedimiento general** es un procedimiento declarado en un módulo (generalmente un módulo estándar). El llamado a este tipo de procedimiento se define explícitamente en el código.

Un **procedimiento asociado a un evento** es un procedimiento que se ejecuta automáticamente ante ciertos eventos de un objeto. Su nombre se forma con el nombre del objeto, seguido del guión bajo "_" y del nombre del evento (ej.: Workbook_Open). El llamado a estos procedimientos es implícito, es decir, el procedimiento se ejecuta automáticamente cuando se produce el evento asociado.

Ejemplo

*El siguiente **procedimiento general** pide al usuario que confirme su deseo de abandonar la aplicación, y sale de Excel si el usuario responde que Sí. Este código puede ejecutarse con un botón de comando o una opción de menú que permita abandonar la aplicación.*

```
Private Sub Terminar()
   If MsgBox("¿Desea salir del programa?", _
      vbQuestion + vbYesNo) = vbYes Then
      Application.Quit
    End If
End Sub
```


Copyrighted Material

*El siguiente **procedimiento asociado a un evento** abre automáticamente el libro Ventas.xls cuando se abre el libro Resumen.xls. Este procedimiento está asociado al evento Open del objeto Workbook y se encuentra en el módulo ThisWorkbook del libro Resumen.xls.*

```
Private Sub Workbook_Open()
'   Abre el libro Ventas.xlsx
   Workbooks.Open Filename:="C:\VENTAS\VENTAS.xlsx"
'   Activa el libro Resumen
   Windows("RESUMEN.xlsx").Activate
End Sub
```

4. Procedimientos Function

Los procedimientos Function, llamados comúnmente **funciones**, devuelven un valor que resulta de un cálculo. El valor se devuelve a través del nombre de la función.

El lenguaje Visual Basic incluye numerosas funciones integradas, tales como las usadas en cálculos con fechas (day, week, year, format, etc.).

Además de estas funciones integradas, usted puede crear sus propias funciones personalizadas.

Ejemplo

La siguiente función pide al usuario que confirme su deseo de abandonar la aplicación y devuelve True si el usuario responde que Sí, y False en caso contrario.

```
Function Terminar() As Boolean
   If MsgBox("¿Desea salir del programa?", _
       vbQuestion + vbYesNo) = vbYes Then
       Terminar = True
   Else
       Terminar = False
    End If
End Function
```

© Editions ENI - All rights reserved

Copyrighted material

5. Declaración de procedimientos

Sintaxis de un procedimiento Sub

```
[Private | Public | Friend] [Static] Sub NomProc
([lista de argumentos])
   <secuencia de instrucciones>
End Sub
```

Sintaxis de un procedimiento Function

```
[Private | Public | Friend] [Static] Function
NomProc ([lista de argumentos]) [As <Type>]
   <secuencia de instrucciones>
End Function
```

Para crear un procedimiento **Sub** o **Function**, se deben respetar los siguientes pasos:

- determine el **alcance** del procedimiento,
- declare el procedimiento según su tipo con la palabra clave **Sub** o **Function**, seguida del nombre del procedimiento,
- defina los **argumentos** que se deban pasar como parámetros al procedimiento e indíquelos entre paréntesis después del nombre del procedimiento,
- en el caso de una función, si es necesario, indique el tipo del valor devuelto después de la palabra clave **As**,
- redacte el código que permita efectuar la operación deseada. Si es necesario, use **Exit Sub** o **Exit Function** para salir del procedimiento. En el caso de una función, asigne el resultado al nombre de la función,
- finalice el procedimiento con **End Sub** o **End Function**.

6. Alcance de los procedimientos

El alcance de un procedimiento determina la extensión de su uso.

Un procedimiento **Public** puede ser llamado desde **todos los módulos** de todos los proyectos Excel.

Copyrighted Material

Un procedimiento **Private** solamente puede ser llamado desde un procedimiento dentro del mismo módulo.

La palabra clave **Static** indica que las variables locales del procedimiento son mantenidas entre un llamado y otro.

Si no se indica Public o Private o Friend, **los procedimientos son públicos por defecto**.

7. Argumentos de los procedimientos

Los argumentos se usan para transferir a los procedimientos parámetros en forma de datos. La cantidad de argumentos puede variar de 0 a varios.

Para declarar un argumento, basta especificar su nombre. Sin embargo, la sintaxis completa para declarar un argumento es la siguiente:

```
[Optional] [ByVal | Byref] [ParamArray]
<variable> [As type]
```

La opción `Optional`	indica que el argumento es opcional y puede omitirse. Los argumentos opcionales deben ubicarse al final de la lista de argumentos, y ser de tipo **Variant**.
La opción `Byval`	indica que el argumento se pasa por valor. El procedimiento accede a una copia de la variable; su valor inicial no se modifica por el procedimiento al que se lo pasa.
La opción `Byref` (opción por defecto)	indica que el argumento se pasa por referencia. En este caso, el procedimiento puede acceder a la variable propiamente dicha; de esta manera, su valor real puede ser modificado por el procedimiento al que lo pasa.
La palabra clave `ParamArray`	se usa únicamente como último argumento de la lista para indicar que se trata de una matriz opcional de elementos de tipo **Variant**. No puede usarse con las palabras clave `ByVal`, `ByRef` u `Optional`.

© Editions ENI - All rights reserved

Copyrighted material

`Variable`	especifica el nombre del argumento. Para las variables de matriz, no especificar su dimensión.
`Type`	especifica el tipo de datos del argumento pasado al procedimiento (**Byte**, **Boolean**, **Integer**, **Long**, etc.).

8. Argumentos con nombre

El pasaje de argumentos a un procedimiento según su orden de aparición es a veces difícil de realizar, especialmente cuando hay parámetros opcionales. De la misma forma, la legibilidad de los llamados a procedimientos con muchos parámetros no siempre es fácil.

Los argumentos con nombre facilitan el pasaje de argumentos gracias a las siguientes ventajas:

- el orden de los argumentos con nombre no es importante.
- los argumentos opcionales pueden omitirse.

La sintaxis de los argumentos con nombres es:

```
NomArg := valor
```

Ejemplo

El siguiente código VBA:

```
If MsgBox("¿Desea salir de la aplicación?", _
   vbYesNo + vbQuestion, "Gestión de ventas") = vbYes Then
   Application.Quit
End If
```

puede transformarse en:

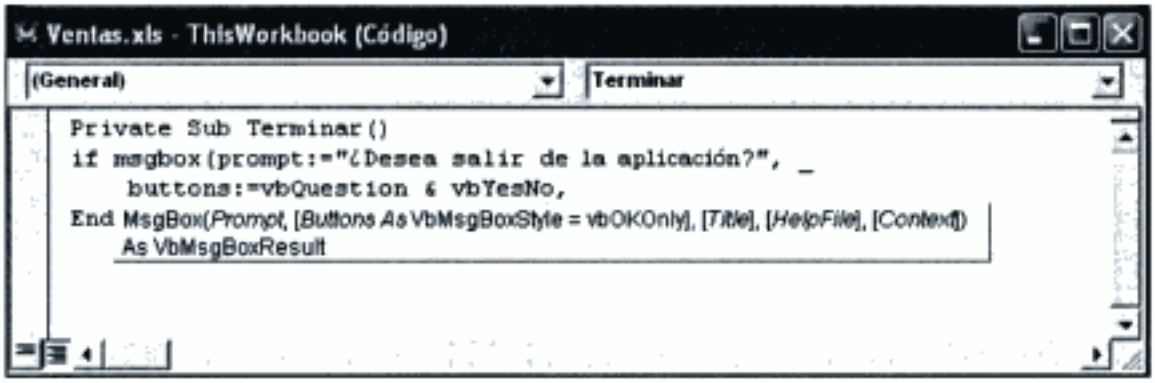
```
If MsgBox(Prompt:= "¿Desea salir de la aplicación?", _
   Buttons:=vbYesNo + vbQuestion, _
   Title:="Gestión de ventas") = vbYes Then
   Application.Quit
End If
```


Copyrighted material

También el orden de los parámetros puede modificarse:

```
If MsgBox(Prompt:= "¿Desea salir de la aplicación?", _
   Title:="Gestión de ventas", _
   Buttons:=vbYesNo + vbQuestion) = vbYes Then
   Application.Quit
End If
```

El nombre de los argumentos aparece automáticamente en el entorno VBE a medida que se escribe la instrucción. Los argumentos opcionales aparecen entre corchetes.



9. Llamar a un procedimiento

Sintaxis

```
[Call] NomProc [lista de argumentos]
```

Si se indica la palabra clave `Call`, debe colocar la lista de argumentos entre paréntesis.

- Para almacenar el resultado de una función en una variable, use la siguiente sintaxis:

  `<variable> = NomProc ( [lista de argumentos] )`

- Para llamar a un procedimiento de otro módulo, use la siguiente sintaxis:

  `NomMódulo.NomProcedimiento`

© Editions ENI - All rights reserved

Copyrighted material

Ejemplo

```
ThisWorkbook.Salir_Apli
```

→ Para llamar a un procedimiento de otro libro, use la siguiente sintaxis:

```
Application.Run "NomLibro!NomMódulo.
NomProcedimiento"
```

Ejemplo

```
Application.Run "Ventas.xls!ThisWorkbook.Salir_Apli"
```

Para ejecutar este comando, el libro Ventas.xls debe estar abierto.

10.Llamar a una función VBA en una fórmula Excel

Las funciones VBA pueden usarse en las fórmulas Excel. Todas las funciones declaradas en **Public** están disponibles en el asistente para funciones de Excel (categoría **Funciones Definidas por el usuario**).

Ejemplo

Este ejemplo usa una función VBA que calcula la edad de una persona a partir de su fecha de nacimiento.

```
Function CalcEdad(fechaNac As Date)
   Dim zFecha As Date
   ' Calcula la edad en función de la fecha de nacimiento
   CalcEdad = Abs(DateDiff("YYYY", fechaNac, Date))
   zFecha = DateAdd("YYYY", CalcEdad,  fechaNac)
   If zFecha > Date Then CalcEdad = CalcEdad - 1
End Function
```

→ Para usar esta función en Excel:

– Seleccione la opción **Insertar función** de la ficha **Formulas**.

Copyrighted material

– En el cuadro de diálogo **Insertar función**, en la lista **O seleccione una categoría**, elija **Definidas por el usuario**; la función CalcEdad estará ahora accesible:

– Seleccione la función y haga clic en **Aceptar**; el cuadro de diálogo le pedirá los argumentos de la función como se ve aquí:

Este ejemplo muestra la importancia del nombre de los argumentos de las funciones: cuanto más explícitos sean éstos, más fácil será usar la función en Excel.

© Editions ENI - All rights reserved

Copyrighted Material

Ahora puede modificar la fórmula para incluir el texto "años" y extender la fórmula a toda la lista.

El resultado en Excel es el siguiente:

E4 =CalcEdad(D4)&" años"

	A	B	C	D	E
1		Fecha de hoy	11-jul-07		
2					
3	Trat.	Apellido	Nombre	Fecha de nacimiento	Edad
4	Srta.	BALDINI	Julia	10/06/1960	47 años
5	Sr.	DURAN	Jacobo	20/11/1982	24 años
6	Sra.	MARTINEZ	Silvia	19/04/1963	44 años
7	Sr.	PEREZ	Jaime	01/12/1945	61 años
8	Srta.	SANCHEZ	Anita	03/07/1972	35 años

11.Ejemplos de procedimientos y funciones

Copiar el contenido de una tabla de valores en la hoja de Excel activa.

```
Sub Mostrar_Tabla()
   Dim TabVal As Variant
   Dim Celda As Range
   Dim i As Integer
   ' Muestra el contenido de la tabla en la hoja de cálculo activa
   TabVal = Array("Buen Día", 1.244, "=A1+12", "=A2+12")
   For i = 0 To 3
      Set Celda = Range("A" & i + 1)
      If MCell(Celda, TabVal(i)) Then
         MsgBox "La celda se actualizó con éxito"
      Else
         MsgBox "La celda no se pudo actualizar"
      End If
   Next i

End Sub
```

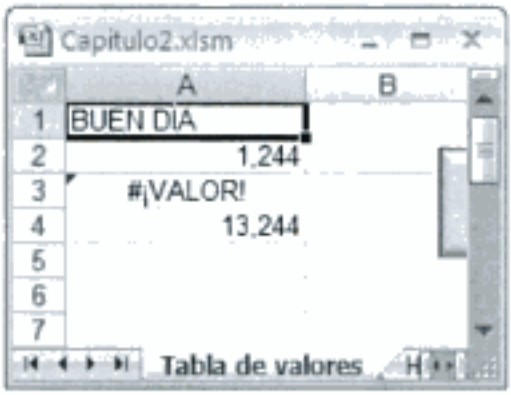
El código de estos ejemplos debe escribirse en un módulo estándar o en el módulo **ThisWorkbook**.

Copyrighted Material

La función MCell informa sobre el valor asignado a una celda. Devuelve True si la celda se actualizó correctamente, y False en caso contrario.

```
Private Function MCell(Celda As Range, _
             Valor As Variant) As Boolean
   ' Actualización de una celda a partir de un valor
   MCell = False
   If Not IsEmpty(Celda) Then Exit Function
   Celda.Value = Valor
   If Celda.Text <> "#VALOR!" Then
      MCell = True
   End If
End Function
```

➔ Si prueba este ejemplo, obtendrá el siguiente resultado:



La tercera celda no se pudo actualizar.

C. Variables

Las variables permiten almacenar valores intermedios durante la ejecución del código VBA para usarlos luego en cálculos, comparaciones, pruebas...

Las variables se identifican por un **nombre** que permite hacer referencia al valor que contienen y un **tipo** que determina la naturaleza de los datos que pueden almacenar.

© Editions ENI - All rights reserved

Copyrighted material

1. Tipos de variables

Numéricas

Tipo	Rango	Tamaño en bytes
Byte	0 a 255	1
Integer (entero)	-32 768 à 32767	2
Long (entero largo)	-2 147 483 648 a 2 147 483 647	4
Single (real simple de coma flotante)	-3,402823E38 a 1,401298E-45 (valores negativos) 1,401298E-45 a 3,402823E38 (valores positivos)	4
Double (real doble de coma flotante)	-1,79769313486231E308 a 4,94065645841247E-324 (valores negativos) 4,94065645841247E-324 a 1,79769313486231E308 (valores positivos)	8
Currency (monetario de punto fijo)	-922 337 203 685 477,5808 a 922 337 203 685 477,5807	8
Decimal	+/-79 228 162 514 264 337 593 543950 335 sin separador decimal; +/-7,92281625142643375935439503 35 con 28 cifras a la derecha del separador decimal; el menor número distinto de cero es +/-0.0000000000000000000000000001	12

Copyrighted material

Cadenas de caracteres

El tipo es **String**. Existen dos tipos de cadenas:

- las cadenas de longitud variable pueden contener aproximadamente dos mil millones de caracteres.
- Las cadenas de longitud fija pueden contener de 1 a aproximadamente 64 Kb de caracteres.

Ejemplo

```
'Cadena de longitud variable
Dim Domicilio As String
'Cadena de longitud fija (20 caracteres)
Dim Nombre As String * 20
```

Boolean o lógica

El tipo es **Boolean**. La variable puede tomar los valores True (Verdadero) o False (Falso), que es su valor por defecto. Ocupa dos bytes.

Fecha

El tipo es **Date**. La variable puede tomar los valores de fecha y de hora del primero de enero del año 100 al 31 de diciembre de 9999. Ocupa ocho bytes.

Variant

Las variables de tipo Variant pueden contener datos de todo tipo, además de los valores especiales **Empty**, **Error** y **Null**.

Usar el tipo de dato Variant ofrece más flexibilidad en el tratamiento de datos. Por ejemplo, si una variable de tipo Variant contiene cifras, se puede usar su valor real o su representación en forma de cadena, según el contexto.

© Editions ENI - All rights reserved

Copyrighted material

De todas formas, las variables de tipo Variant requieren 16 bytes de memoria para números y 22 bytes + la longitud de la cadena para los caracteres; esto puede ser perjudicial en el caso de procedimientos largos o en módulos complejos.

Ejemplo

```
Sub Variable_Variant()
'Declaración de la variable "Valx" como Variant
    Dim Valx As Variant
'Asignación de una sucesión de valores a la variable
'y mostrar el tipo del resultado

'10 da Integer
    Valx = 10
    MsgBox Valx & " es de tipo " & TypeName(Valx)
'Ejemplo da String
    Valx = "Ejemplo"
    MsgBox Valx & " es de tipo " & TypeName(Valx)
'Esta multiplicación da Double
    Valx = 12500.32 * 1E+21
    MsgBox Valx & " es de tipo " & TypeName(Valx)
'#1/1/99# da Date
    Valx = #1/1/99#
    MsgBox Valx & " es de tipo " & TypeName(Valx)
'True da Boolean
    Valx = True
    MsgBox Valx & " es de tipo " & TypeName(Valx)

End Sub
```

Objeto

El tipo es **Object**. Para crear una variable que contenga un objeto, comience por declarar la variable como tipo Objeto y luego asígnele un objeto.

→ Para declarar una variable Objeto:

si el tipo de objeto no se conoce, use la sintaxis:

`InstrucciónDeDeclaración NomVariable` **`As Object`**

si se conoce el tipo de objeto, use la sintaxis:

`InstrucciónDeDeclaración NomVariable` **`As TipoObjeto`**

Copyrighted Material

Ejemplo

```
Sub Variables_Objeto()
'Test se declara como objeto
'NomCli se declara como hoja de cálculo
'AImprimir se declara como gráfico
    Dim Test As Object
    Dim NomCli As Worksheet
    Dim AImprimir As Chart
End Sub
```

- Para asignarle un objeto a una variable Objeto, use la instrucción **Set**:

```
Set NomVariable = ObjetoaAsignar
```

Ejemplo

Declaración de una variable ZonaTest destinada a contener un objeto Range y asignación de las celdas A6 a B15 a esta variable:

```
Dim ZonaTest As Range
Set ZonaTest = Range("A6:B15")
```

- Para finalizar la asociación entre una variable y un objeto determinado, use la siguiente sintaxis:

```
Set NomVariable = Nothing
```

Definido por el usuario (o personalizado)

Los tipos de datos personalizados se crean con la instrucción **Type** usada a nivel de módulo.

Sintaxis

```
Type NomTipoPerso
      NomElemento1 As TipoDatos
      NomElemento2 As TipoDatos
      ...
End Type
```

La definición del tipo solamente puede hacerse en la sección de declaración de un módulo.

© Editions ENI - All rights reserved

Copyrighted material

Ejemplo

Declaración de un tipo personalizado constituido por una letra y un número entero en el módulo de código ProGene.

Capitulo2.xlsm - ProGene (Código)

(General) | CalcEdad

```
Option Explicit
    ' Creación de un tipo personalizado constituido
    ' por una letra y un número entero
    Type TDCodArticulo
        Letra As String * 1
        Numero As Integer
    End Type
```

Uso del tipo personalizado para controlar los códigos de artículos introducidos de la celda B6 a la celda B11 de la hoja de cálculo activa. En caso de error, se muestra un mensaje.

Capitulo2.xlsm - ThisWorkbook (Código)

(General) | Ctrl_Articulos

```
Sub Ctrl_Articulos()
    Dim CodArt As TDCodArticulo
    Dim ZonaTest As Range
    Dim Cell As Range

    '   Controla los códigos introducidos
    Set ZonaTest = Range("B6:B11")
    On Error GoTo Error1:
    For Each Cell In ZonaTest
        CodArt.Letra = Cell.Characters(1, 1).Text
        CodArt.Numero = Cell.Characters(2).Text
1   Next
    Exit Sub

'   Procedimiento de administración de errores
Error1:
    Cell.Select
    MsgBox "Código de artículo incorrecto: " & Cell
       Resume 1

End Sub
```


Copyrighted Material

2. Declaración de variables

Para crear una variable, debe declararla, es decir, darle un nombre. Luego puede usar ese nombre para modificar el valor de la variable, usar ese valor, etc.

La declaración de variables en VBA puede ser **implícita** o **explícita**.

a. Declaraciones implícitas

Se hacen directamente al asignar un valor a un nombre de variable. El tipo de datos será entonces el tipo por defecto, o sea, **Variant**.

Ejemplo

```
I = 12
Importe = 12000
Nombre = "Juan Carlos"
```

b. Declaraciones explícitas

Requieren el uso de una **instrucción de declaración** (Dim, Public, Private, etc.). Si el tipo de la variable no se indica, la variable resultará del tipo por defecto, o sea, **Variant**.

Se puede imponer la declaración implícita de variables usando la instrucción **Option Explicit** en la sección de declaración de cada módulo. Para insertar esta instrucción automáticamente en cada nuevo módulo, active la opción **Requerir declaración de variables** del menú **Herramientas** - **Opciones** - ficha **Editor**.

Ejemplo

```
Dim I
Private Importe As Double
Public Nombre as String
```

© Editions ENI - All rights reserved

Copyrighted Material

Para optimizar la velocidad de ejecución del código VBA, se recomienda declarar las variables en forma explícita.

c. Sintaxis de las instrucciones de declaración

```
<InstrucciónDeDeclaración> NomVariable [As <TipodeDatos>]
```

Donde `InstrucciónDeDeclaración` corresponde a una de las siguientes cuatro instrucciones: `Dim`, `Public`, `Private` o `Static`.

`Dim` Las variables declaradas con la instrucción `Dim` **a nivel de módulo** están disponibles para todos los procedimientos del módulo. No son accesibles desde ningún otro módulo.
Las variables declaradas con la instrucción `Dim` **a nivel de procedimiento** solamente están disponibles dentro del procedimiento.

`Private` **Solamente a nivel de módulo**. Las variables `Private` solamente están disponibles para el módulo en el que son declaradas.

`Public` **Solamente a nivel de módulo**. Las variables declaradas con la instrucción `Public` son accesibles desde **el conjunto de módulos** de todos los proyectos Excel abiertos.
Si la instrucción `Option Private Module` se especifica en la sección de declaración del módulo, las variables solamente son públicas dentro del proyecto que las recibe.

`Static` **Solamente a nivel de procedimiento**. Las variables declaradas con la instrucción `Static` conservan su valor durante toda la duración de la ejecución del código.

Copyrighted material

3. Declaración de los tipos de variables

a. Declaraciones explícitas del tipo

El tipo de la variable se especifica en la declaración, tras la palabra clave **As**.

Ejemplo

```
Sub TotalAcum()
     Dim Total As Integer
     Static Acum As Integer
     Total = Total + 10
     Acum = Acum + 10
'Devuelve 10 en cada ejecución del procedimiento
     MsgBox Total
'Devuelve 10 en la primera ejecución, luego 20 la segunda vez,
'30 la tercera, etc.
     MsgBox Acum
End Sub
```

Usted puede declarar más de una variable en una misma instrucción pero, atención, el tipo de datos solamente será tenido en cuenta para la última variable, el tipo Variant se asignará a las otras.

Ejemplo

La siguiente instrucción declara la variable Identificador de tipo Variant y las variables Superficie y Latitud de tipo entero.

```
Dim Identificador, Superficie As Integer
Dim Latitud As Integer
```

b. Declaraciones implícitas del tipo

El tipo de variable se declara por el uso de un sufijo en el momento de su utilización o por la instrucción **DefType**.

© Editions ENI - All rights reserved

Copyrighted Material

Empleo de un sufijo

Usted debe agregar uno de los siguientes caracteres al nombre de la variable:

Sufijo	Tipo de datos
%	Integer
&	Long
!	Single
#	Double
@	Currency
$	String

Ejemplo

Declara la variable como tipo Cadena (String).

```
Dim Nombre$
```

Declara la variable como tipo Monetario (Currency).

```
Dim Deuda@
```

DefType

Estas instrucciones se emplean en la zona de declaración del módulo para definir los tipos de datos por defecto de las variables cuyos nombres comienzan por los caracteres especificados.

Lista de instrucciones **DefType**:

Instrucción	Tipo de datos
DefBool	Boolean
DefDbl	Double
DefInt	Integer
DefDate	Date
DefLng	Long

Copyrighted material

Instrucción	Tipo de datos
DefStr	String
DefCur	Currency
DefObj	Object
DefSng	Single
DefVar	Variant
DefByte	Byte

Ejemplo

Todas las variables cuyos nombres comienzan por una letra comprendida entre I y K y por la letra N son variables de tipo entero (Integer).

```
DefInt I-K,N
```

Las variables que comienzan por una letra comprendida entre A y H serán de tipo cadena (String).

```
DefStr A-H
```

4. Matrices

Usted puede crear una variable matriz cuando necesite trabajar con un grupo de valores relacionados.

➔ Para crear una variable matriz, use la siguiente sintaxis:

```
<InstrucciónDeDeclaración> NomMatriz(índices) [As<TipodeDatos>]
```

Donde para **(índices)**:

- Si se omite este argumento: la matriz tendrá dimensión libre.
- Si se indica un valor: la matriz tendrá un número determinado de elementos.
- Si se indica `LimiteInf To LimiteSup`: la matriz tendrá un número de elementos determinados y números de índice específicos.

Por defecto, el menor valor del índice de una matriz es 0.

© Editions ENI - All rights reserved

Copyrighted material

- Para modificar el menor valor del índice, use la instrucción **`Option Base`** `MenorValorÍndice` en la sección de declaración del módulo o use la sintaxis `LimiteInf` **`To`** `LimiteSup` en el argumento **`(índices)`**.

- Para definir el contenido de una matriz, use la función **Array** (la variable debe ser una tabla de dimensión libre) o despliegue los datos en una hoja o asigne una por una cada variable de la matriz usando los índices.

Ejemplo

El siguiente ejemplo muestra, en la hoja activa de Excel, la lista de los factores de conversión a euros para cinco países.

```
Const FacFRF = 6.55957
Const FacBEF = 40.3399
Const FacDEM = 1.95583
Const FacESP = 166.386
Const FacITL = 1936.27
```

```
Sub Muestra_Factor()
   Dim Pais As Variant
   Dim Factor(5) As Double
   Dim i As Integer
   ' Lista de países
   País = Array("Francia", "Bélgica", "Alemania", _
           "España", "Italia")

   ' Lista de los factores por país
   Factor(0) = FacFRF
   Factor(1) = FacBEF
   Factor(2) = FacDEM
   Factor(3) = FacESP
   Factor(4) = FacITL

   ' Muestra los países y sus factores en la hoja de cálculo Euro
   For i = 0 To 4
        With Sheets ("Euro")
              .Cells(i + 1, 1) = Pais(i)
              .Cells(i + 1, 2) = Factor(i)
        End With
   Next i

End Sub
```


Copyrighted Material

En este ejemplo, es igualmente posible usar una matriz de dos índices. El código VBA del procedimiento queda como sigue:

```
Sub Muestra_Factor2
' Matriz de dos índices
   Dim TabFactor(5, 1)
   Dim i As Integer

   ' Lista de países y sus factores
   TabFactor(0, 0) = "Francia"
   TabFactor(0, 1) = FacFRF
   TabFactor(1, 0) = "Bélgica"
   TabFactor(1, 1) = FacBEF
   TabFactor(2, 0) = "Alemania"
   TabFactor(2, 1) = FacDEM
   TabFactor(3, 0) = "España"
   TabFactor(3, 1) = FacESP
   TabFactor(4, 0) = "Italia"
   TabFactor(4, 1) = FacITL

   ' Muestra los países y sus factores en la hoja de cálculo Euro
   For i = 0 To 4
        With Sheets ("Euro")
             .Cells(i + 1, 1) = TabFactor(i, 0)
             .Cells(i + 1, 2) = TabFactor(i, 1)
        End With
   Next i

End Sub
```

5. Constantes

Una constante permite asignar un nombre explícito a un valor.

a. Constantes personalizadas

La declaración de una constante se hace con la instrucción **Const** en la sección de declaración de un módulo o un procedimiento.

```
Const NomConstante [As <TipoDato>] = <expresión>
```

© Editions ENI - All rights reserved

Copyrighted material

`As TipoDato`	el tipo de datos no puede ser un objeto (Object) ni un tipo personalizado (Type).
`expresión`	no puede ser una función definida por el usuario, ni una función intrínseca de Visual Basic.

Ejemplo

Declaración de algunas constantes.

```
Sub Constantes()
   'Ejemplos de constantes autorizadas:
    Const Val1 = "Mega+"
    Const Val2 = 148
    Const Val3 = 125.45
   'El tipo de datos es Double
    Const Val4 As Single = 125.45
   'Permite ahorrar memoria
     Const Val5 = Val2 * Val3
     Const Val6 = Val1 & " cuesta " & Val2
   'Ejemplo de constante no autorizada:
   'Por el uso de una función VB
     Const Val6 = Fix(Val4)
End Sub
```

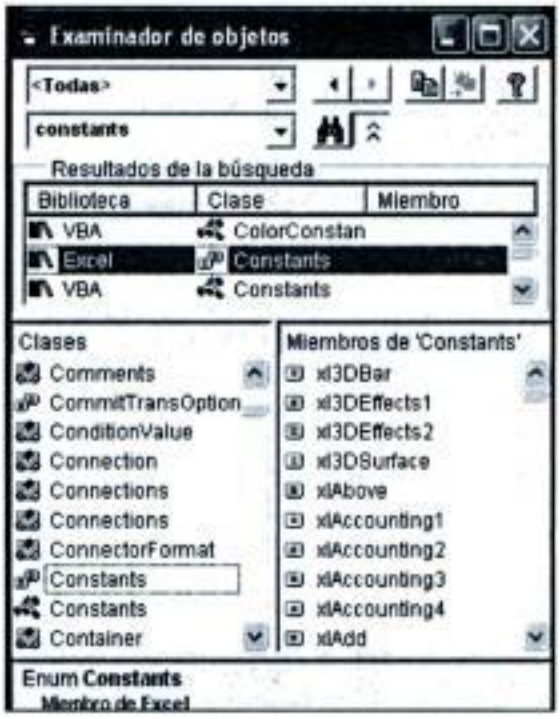
Para crear una constante accesible al conjunto de los libros, se la debe declarar en la sección de declaración de un módulo y ubicar la instrucción **Public** antes de la instrucción **Const**.

b. Constantes integradas

Las constantes usadas por los objetos Microsoft Excel van precedidas por las letras "xl"; las constantes usadas con otras instrucciones y funciones Visual Basic van precedidas por las letras "vb", y las constantes Microsoft Office van precedidas por las letras "mso".

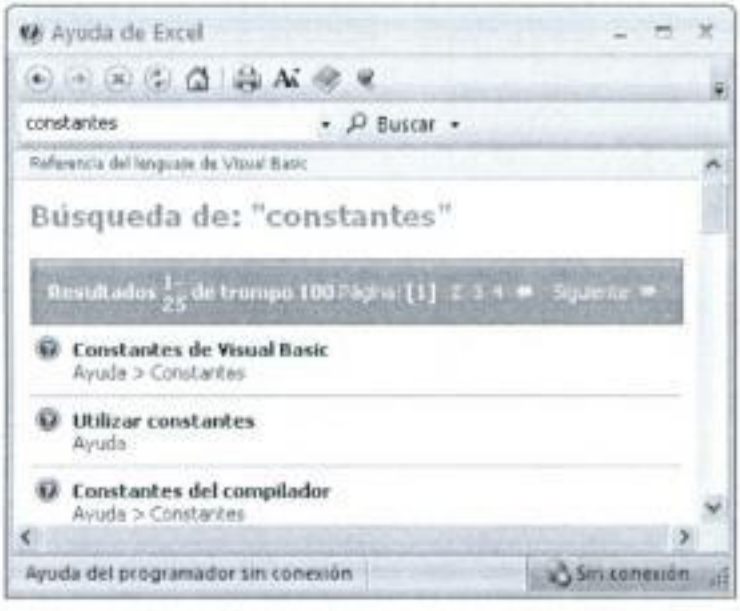
→ Para mostrar la lista de constantes integradas, abra el examinador de objetos con un clic en el icono o pulsando la tecla de función [F2]. Escriba la palabra **Constants** en la lista desplegable **Texto de búsqueda** y luego haga clic en el icono .

Copyrighted material



→ Para obtener la lista de constantes Visual Basic, proceda de la siguiente manera:

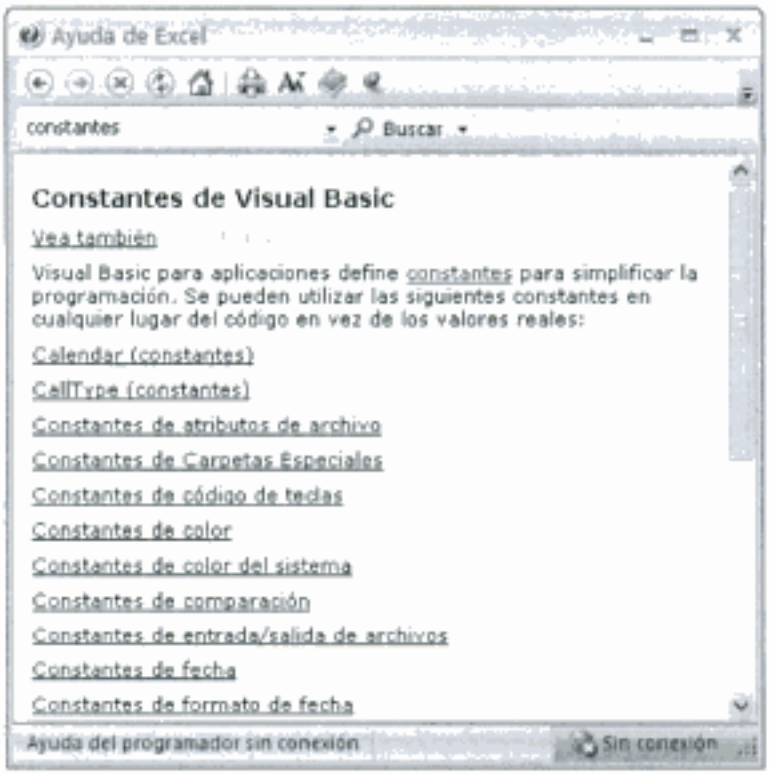
- Abra la ayuda de Visual Basic por medio del icono de la barra de herramientas **Estándar** o de la tecla [F1],
- Escriba la palabra "Constantes" en la zona de búsqueda y haga clic en el botón **Buscar**:



© Editions ENI - All rights reserved

Copyrighted material

– La lista de constantes clasificadas por tipo estará ahora accesible:



D. Estructuras de decisión

Es conveniente testear las condiciones específicas antes de ejecutar las instrucciones.

Las estructuras de decisión, llamadas también alternativas o bifurcaciones condicionales, permiten, tras una evaluación, optar por uno u otro bloque de código.

Se distinguen dos instrucciones de bifurcación condicional:

– If ... Then ... Else

– Select ... Case

La función IIf también puede usarse para definir un valor en función de una condición. Ejemplo: Port = IIf(Cantidad < 100, 100, 0).

Copyrighted material

1. Instrucción IF

Permite ejecutar ciertas instrucciones en función del resultado de una condición.

➔ If...Then

```
If <condición> Then <instrucción> [:<instrucción>]
```

Si hay varias instrucciones, sepárelas por el signo de puntuación **:** (dos puntos). Esta sintaxis se usa especialmente para pruebas cortas y simples.

Ejemplo

Si la celda A1 está vacía, emita un bip y muestre un mensaje.

```
Sub Test_Celda_A1()
  If IsEmpty(Range("A1")) Then Beep: MsgBox "Olvidó el título"
End Sub
```

➔ If...Then...End If

```
If <condición> Then
   <instrucción1>
   <instrucción2>
...
End If
```

Ejemplo

```
Sub Test_Titulo()
'Si la celda A1 no está vacía
'entonces ponerla en negrita y pintarla de rojo
    If Not IsEmpty(Range("A1")) Then
        With Range("A1")
            .Font.Bold = True
            .Interior.ColorIndex = 3
        End With
    End If
End Sub
```

© Editions ENI - All rights reserved

Copyrighted material

➔ If...Then...Else...End If

```
If <condición> Then
   <instrucciones>
Else
   <instrucciones>
End If
```

Ejemplo

Al cambiar la moneda (euro o US$) de la celda C3, modificar el formato del rango D6:F11.

```
Private Sub Workbook_SheetChange(ByVal Sh As Object, _
ByVal Target As Range)
'  Modifica la celda C3 de la hoja Artículos
If Sh.Name = "Artículos" And Target.Address = "$C$3" Then
   Aplicar_Formato
End If
End Sub
```

Este procedimiento modifica el formato de las celdas en función de la moneda elegida.

```
Sub Aplicar_Formato()
'   Formato € o $
Range("D6:F11").NumberFormat = "0.00"
If UCase(Range("C3")) = "EURO" Then
   Range("D6:F11").NumberFormat = "0.00 €"
Else
   Range("D6:F11").NumberFormat = "0.00"" $"""
End If
End Sub
```


Copyrighted material

➔ If...Then...ElseIf...Else...End If

```
If <condición> Then
   <instrucciones>
ElseIf <condición> Then
   <instrucciones>
ElseIf <condición> Then
   <instrucciones>
...
          ...
Else
   <instrucciones>
End If
```

Ejemplo

Este procedimiento modifica los textos de las celdas seleccionadas: si la última letra es una minúscula, pasa todo a mayúsculas; si no, pasa todo a minúsculas con la primera letra en mayúscula.

```
Sub Mayus_Minus()
   Dim Celda As Range
   Dim CodAscii As String

   '   Recorre las celdas de la selección
   For Each Celda In Selection
      If IsEmpty(Celda) Then
          Beep
          MsgBox "La celda " & Celda.Address & " está vacía"
      Else
         '   Código Ascii de la última letra
         CodAscii = Asc(Right(Celda, 1))
         '   Si está en mayúscula
         If CodAscii >= 65 And CodAscii <= 90 Then
            Celda.Value = UCase(Left(Celda, 1)) _
                                & LCase(Right(Celda, Len(Celda) - 1))
         '   Si está en minúscula
         ElseIf CodAscii >= 97 And CodAscii <= 122 Then
               Celda.Value = UCase(Celda)
         Else                                                  .../...
```

© Editions ENI - All rights reserved

Copyrighted Material

```
.../...
                MsgBox "El último carácter de la celda: " _
                & Celda.Address & " no es una letra"
            End If
        End If
    Next
End Sub
```

2. Instrucción Select Case

➔ Select Case

Ejecuta una secuencia de instrucciones específicas en función del valor de una expresión.

```
Select Case <ExpresiónTest>
   Case <ListaExpresiones>
        <instrucciones>
   Case <ListaExpresiones>
        <instrucciones>
...
    ...
   Case Else
        <instrucciones>
End Select
```

`<listaExpresiones>` puede tomar una de las siguientes formas:

- valor (ejemplo: `Case 10`)
- lista de valores (ejemplo: `Case 1, 5, 10`)
- rango de valores (ejemplo: `Case 1 To 5`)
- expresión condicional (ejemplo: `Case Is >= 5`)

Copyrighted Material

Ejemplo

Llamado a un procedimiento que determina la fórmula de cálculo del total en función de la cantidad, del precio y del flete.

```
Private Sub Calc_Total()
   Dim ACalcular As Range
   Dim Cell As Range

   '   Llama a la función Total usando como parámetros
   '   las celdas Cant, Precio y Flete
   Range("D2") = Total(Range("A2"), Range("B2"), Range("C2"))
End Sub
```

La función Total devuelve una fórmula Excel. El flete es gratuito a partir de dos unidades ordenadas, el porcentaje de descuento también depende de la cantidad ordenada.

```
Function Total(zCant As Range, ZPrecio As Range, zFlete As Range) _
As String

   '   Determinación de la fórmula de cálculo del total
   '   en función de la cantidad ordenada
   Select Case zCant
      Case 1
          Total = "=" & ZPrecio.Address & "+" & zFlete.Address
      Case 2 To 10
          Total = "=" & zCant.Address & "*" & ZPrecio.Address
      Case 11 To 100
          Total = "=" & zCant.Address & "*" & ZPrecio.Address & "* 0.95"
      Case 101 To 1000
           Total = "=" & zCant.Address & "*" & ZPrecio.Address & "* 0.9"
      Case Else
           Total = "Error en la cantidad"
   End Select
End Function
```

© Editions ENI - All rights reserved

Copyrighted material

E. Estructuras en ciclo

Las estructuras en ciclos (o repetitivas) permiten repetir la ejecución de un conjunto de instrucciones.

Se distinguen varios tipos de estructuras en ciclo:

- Do...Loop
- While...Wend
- For...Next
- For Each...Next

Do...Loop y **While...Wend** repiten las operaciones en función de una cierta condición, mientras que **For...Next** repite las operaciones una cantidad de veces dada por un contador.

For Each...Next permite recorrer los elementos de una colección.

1. Instrucción Do...Loop

Ejecuta un bloque de instrucciones un número indeterminado de veces.

Sintaxis 1

Las instrucciones se ejecutan mientras que la condición devuelve el valor True.

```
Do While <Condición>
    <Instrucciones>
Loop
```

Sintaxis 2

Las instrucciones se ejecutan una primera vez sin condición y luego mientras la condición devuelva True.

```
Do
    <Instrucciones>
Loop While <Condición>
```


Copyrighted Material

Ejemplo:

El siguiente código solicita al usuario que escriba un número mientras que el valor introducido no sea numérico o superior a 100.

```
Sub Introducir_Numero ()
   Dim strRep
   Do
      strRep = InputBox("Introduzca un número > 100")
   Loop While (Not IsNumeric(strRep) Or strRep <= 100)
End Sub
```

Sintaxis 3

Las instrucciones se ejecutan hasta que la condición toma el valor True (mientras que la condición devuelva el valor False).

```
Do Until <Condición>
    <Instrucciones>
Loop
```

Sintaxis 4

Las instrucciones se ejecutan una primera vez sin condición y luego hasta que la condición devuelva el valor True.

```
Do
    <Instrucciones>
Loop Until <Condición>
```

Ejemplo

El siguiente código solicita al usuario que escriba un número hasta que el valor introducido sea numérico y mayor que 100.

```
Sub Introducir_Numero()
   Dim strRep
   Do
      strRep = InputBox("Introduzca un número > 100")
   Loop Until (IsNumeric(strRep) And strRep > 100)
End Sub
```

© Editions ENI - All rights reserved

Copyrighted material

2. Instrucción While...Wend

Ejecuta una serie de instrucciones en un ciclo mientras se cumple la condición especificada.

Sintaxis

```
While <condición>
    <instrucciones>
Wend
```

Ejemplo

```
Sub Introducir_Precio()
   '   Pide la introducción de un precio mientras que
   '   sea vacío o incorrecto
   While IsEmpty(Range("C9")) Or Not IsNumeric(Range("C9"))
      Range("C9") = InputBox("Introduzca el precio del producto")
   Wend
End Sub
```

3. Instrucción For...Next

Ejecuta un bloque de instrucciones según el valor de un contador.

```
For <contador>=<inicio> To <fin> [Step <incremento>]
    <instrucciones>
Next
```

Ejemplo

Este procedimiento inserta los totales trimestrales en una matriz de resultados mensuales; además, el procedimiento elimina los totales si ya los había.

```
Sub Totales_Trimestrales()
   Dim nTrim As Integer
   Dim i As Integer
   Dim Celda as Range                                    .../...
```


Copyrighted Material

```
.../...
   '   Inserta los totales trimestrales cada 3 meses,
   '   los pone en negrita con un borde
   nTrim = 1
   For i = 5 To 17 Step 4
       If Left(Cells(2, i), 4) <> "Trim" Then
          Cells(2, i).EntireColumn.Insert
          Cells(2, i) = "Trim. " & nTrim
          nTrim = nTrim + 1
          Range(Cells(3, i), Cells(11, i)).Select
          Selection.FormulaR1C1 = "=SUM(RC[-3]:RC[-1])"
          Range(Cells(2, i), Cells(11, i)).Font.Bold = True
          For Each Celda In Range(Cells(2, i), Cells(11, i))
              CeldaBorderAround ColorIndex:=1, Weight:=xlThin
          Next Celda
      End If
      Range("A1").Activate
   Next
End Sub
```

```
Sub Suprime_Totales_Trimestrales()
   Dim i As Integer

   '   Suprime los totales trimestrales si ya los había
   For i = 5 To 17
         If Left(Cells(2, i), 4) = "Trim" Then
            Cells(2, i).EntireColumn.Delete
         End If
   Next i
End Sub
```

Este procedimiento muestra en la hoja de cálculo "Colores" los diferentes colores de relleno y el valor de la propiedad ColorIndex correspondiente.

```
Sub Muestra_Colores()
   Dim i As Integer
      With Sheets("Colores")
           For i = 1 To 56
               Cells(i, 1).Interior.ColorIndex = i
               Cells(i, 2) = i
           Next i
       End With
End Sub
```

© Editions ENI - All rights reserved

Copyrighted material

4. Instrucción For Each...Next

Ejecuta un bloque de instrucciones para cada elemento de una colección de objetos o de una matriz.

```
For Each <elemento> In <Grupo>
    <Instrucciones>
Next <elemento>
```

Ejemplo

Estos procedimientos aplican un color de letra a las celdas en función de su contenido.

```
Sub Colores_Celda()
   Dim ZonaaModificar As Range
   Dim Celda As Range

' Aplica un color en función del valor de la celda
   Set ZonaaModificar = Range("B3:Q11")
   For Each Celda In ZonaaModificar
        Select Case Celda
             Case Is < 1000
                  Celda.Font.Color = RGB(150, 150, 250)
             Case Is < 5000
                  Celda.Font.Color = RGB(90, 100, 250)
             Case Is < 10000
                  Celda.Font.Color = RGB(10, 20, 250)
             Case Is < 20000
                  Celda.Font.Color = RGB(5, 10, 175)
             Case Else
                  Celda.Font.Color = RGB(5, 5, 100)
        End Select
   Next
End Sub
```

Ejecución de varias acciones sobre un objeto

```
With objeto
    <Instrucciones>
End With
```


Copyrighted Material

Ejemplo

```
Sub Paginacion()
'  Define la paginación de la hoja activa
'  Redimensiona las columnas y procede con la impresión
   With ActiveSheet
      With .PageSetup
            .Orientation = xlLandscape
            .LeftMargin = Application.InchesToPoints(0.5)
            .RightMargin = Application.InchesToPoints(0.5)
            .TopMargin = Application.InchesToPoints(0.5)
            .BottomMargin = Application.InchesToPoints(0.5)
            .LeftHeader = ""
            .CenterHeader = "&A"
            .RightHeader = ""
            .LeftFooter = ""
            .CenterFooter = "Page &P"
            .RightFooter = ""
       End With
       .Columns("A:Q").EntireColumn.AutoFit
       .PrintOut
   End With
End Sub
```

5. Salir de las estructuras de control

La instrucción **Exit For** permite salir directamente de un ciclo **For** o **For Each**, mientras que **Exit Do** sale directamente de un ciclo **Do**.

Ejemplo

```
Sub Introducir_Fecha()
   Dim strVal
   '  Fuerza la introducción de una fecha en la celda A1
   '  Si no se introduce ningún valor: se sale del ciclo
   Range("A1") = ""
   Do While Not IsDate(Range("A1"))
        strVal = InputBox("Escriba una fecha")
        If strVal <> "" Then
           If IsDate(strVal) Then Range("A1") = strVal
        Else
            Exit Do
        End If
   Loop
End Sub
```

© Editions ENI - All rights reserved

Copyrighted material

F. Operadores

Los operadores permiten realizar operaciones aritméticas con variables y/o constantes, comparar variables entre ellas, evaluar varias condiciones, etc.

Se distinguen varios tipos de operadores:

- operadores aritméticos,
- operadores de comparación,
- operadores lógicos,
- operador de concatenación.

El operador de asignación es el signo =. El valor de la expresión situada a la derecha del signo igual es asignado a la variable situada a la izquierda del signo (ejemplo: IntA = 12, IntA = Intb * 12).

1. Operadores aritméticos

Permiten efectuar cálculos aritméticos con variables y/o constantes.

Operador	Cálculo realizado
+	Adición
-	Sustracción
/	División con resultado de un número con coma flotante
Mod	Resto de la división entre dos números
\	División con resultado entero
*	Multiplicación
^	Potenciación

Copyrighted material

2. Operadores de comparación

Comparan dos valores o dos cadenas de caracteres.

Operador	Cálculo realizado
<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que
=	Igual a
<>	Distinto de

La instrucción **Option Compare** utilizada a nivel de módulo permite declarar el método de comparación por defecto que conviene usar en la comparación de cadenas. Puede tomar tres valores posibles:

- La opción **Compare Binary** (opción por defecto) realiza la comparación de cadenas basada en el orden derivado de la representación binaria interna de los caracteres: A < B < E < Z < a < b < e < z < À < Ê < Ø < à < ê...
- La opción **Compare Text** realiza la comparación de cadenas sin distinguir mayúsculas de minúsculas: (A=a) < (À=à) < (B=b) < (E=e) < (Ê=ê) < (Z=z) < (Ø=ø)...
- La opción **Compare Database** realiza la comparación de cadenas basado en el orden determinado por el identificador de parámetros regionales de la base de datos en la que se realiza la comparación de cadenas.

3. Operadores lógicos

Permiten evaluar simultáneamente dos (o más) valores booleanos o expresiones que devuelven este tipo de valor. Generalmente se usan con la instrucción IF.

© Editions ENI - All rights reserved

Copyrighted material

Operador	Cálculo realizado
AND	Si todas las expresiones tienen el valor True, el resultado es True. Si una de las expresiones tiene el valor False, el resultado es False.
OR	Si por lo menos una de las expresiones tiene el valor True, el resultado es True (o inclusivo).
XOR	Si una y sólo una de las expresiones tiene el valor True, el resultado es True (o exclusivo).
NOT	Devuelve el contrario de la expresión.
Eqv	Devuelve True si las dos expresiones son idénticas.

Ejemplo

```
(A>=1) AND (A=<9) devuelve True si A está comprendido entre 1 y 9,
NOT (A >= 10) devuelve True si A es estrictamente menor que 10,
(A>0) OR (B>0) OR (C>0) devuelve True si al menos uno de los valores
es positivo.
```

4. Operador de concatenación

El operador de concatenación es el signo **&**. Engancha cadenas de caracteres, valores y expresiones.

Ejemplo

Concatenación del apellido y el nombre.

```
StrApeNom = Apellido & " " & Nombre
```

5. Prioridad de los operadores

Cuando hay varios operadores en una misma expresión, cada uno de ellos se evalúa en un orden predeterminado, llamado prioridad de los operadores.

Copyrighted material

Los operadores se evalúan en el siguiente orden: operadores aritméticos, operadores de comparación, operadores lógicos.

Los operadores de comparación tienen la misma prioridad; es decir, son evaluados por orden de aparición, de izquierda a derecha.

Los operadores aritméticos y lógicos se evalúan en el siguiente orden de prioridad:

Aritmético	**Lógico**
^	Not
*, /, \	And
Mod	Or
+, -	Xor
	Eqv

El uso de paréntesis permite modificar la prioridad para que un elemento de una expresión sea evaluado antes que los otros. Las operaciones encerradas entre paréntesis se evalúan siempre antes que las otras.

Ejemplo

```
La expresión "3 + 4 * 5" da como resultado 23.
La multiplication (4 * 5) se efectúa antes que la adición ( + 3)
La expresión "(3 + 4) * 5" devuelve 35. La adición se efectúa con
prioridad.
```

Se aconseja usar paréntesis para mejorar la legibilidad del código.

© Editions ENI - All rights reserved

Copyrighted Material

G. Reglas de escritura del código

1. Comentarios

Los comentarios permiten documentar el código VBA para hacerlo más legible.

```
REM comentario
```

o

```
' comentario
```

Al validar la línea de comentario, ésta se muestra, por defecto, en verde.

2. Carácter de continuación

Una instrucción VBA puede escribirse en muchas líneas usando un guión bajo "_" precedido de un espacio.

Ejemplo

```
'   Pide la introducción de un precio en tanto que
'   esté vacío o sea incorrecto

Do While IsEmpty(Precio) Or Not IsNumeric(Precio) _
   Or Precio < 50 Or Precio > 500
   Precio = InputBox("Escribir un importe comprendido entre " _
          & "50 y 500 ")
Loop
```

3. Sangrías

Las sangrías (o tabulaciones) permiten una mayor legibilidad del código. Es especialmente importante usarlos en las estructuras de control (sobre todo si hay varias instrucciones If anidadas) y las estructuras de decisión.

- Para generar las sangrías, use la tecla [Tab].
- Para retroceder a la tabulación precedente, use las teclas [Mayús] [Tab].

Copyrighted Material

➲ Para modificar el tamaño de la tabulación (cuatro espacios por defecto), seleccione **Opciones** en el menú **Herramientas**, haga clic en la ficha **Editor** y modifique el valor **Ancho de tabulación**.

4. Nombres de procedimientos, variables y constantes

Los nombres de los procedimientos, constantes, variables y argumentos deben respetar las siguientes reglas:

- el primer carácter debe ser una letra,
- no se diferencian minúsculas de mayúsculas (se aceptan letras acentuadas), aunque se respetan unas y otras,
- no se usan nombres reservados a Visual Basic o palabras clave con restricciones,
- no se usan el punto, el espacio ni los signos !, $, # y @,
- un nombre no puede tener más de 255 caracteres,
- para los procedimientos **Function**, no se usa un nombre igual a una referencia de celda,
- no se indican varias veces los mismos nombres de variables y de constantes en un mismo nivel de alcance.

© Editions ENI - All rights reserved

Copyrighted material

Copyrighted Material

Capítulo 3: La programación de objetos en Excel

A. Presentación 86

B. El modelo de objetos de Excel 87

1. Presentación 87
2. Objetos y colecciones 88

C. Principios de uso de los objetos y las colecciones . . 91

1. Propiedades 91
2. Propiedades que representan objetos 91
3. Métodos 93
4. Eventos 94
5. Colecciones 95
6. Redacción automática de instrucciones 97

D. Instrucciones usadas con los objetos 99

1. La instrucción With 99
2. La instrucción For Each... Next 99
3. La instrucción If TypeOf 100
4. La instrucción Set 101

E. El Examinador de objetos 102

1. Presentación 102
2. Búsqueda en el Examinador de objetos 104

Copyrighted material

A. Presentación

VBA Excel es un lenguaje de programación **orientado a objetos**, si bien no dispone de todas las funcionalidades de los lenguajes de este tipo.
La mayoría de los elementos que maneja Excel son objetos: los libros, las hojas de cálculo, los rangos de celdas, las celdas, etc.

Los objetos se organizan según un **modelo jerárquico**: ciertos objetos contienen otros objetos que pueden, a su vez, contener otros. Estos objetos se llaman **contendedores** u **objetos Parent**. Por ejemplo, el objeto **Application** es un contenedor de objetos **Workbook** (libros abiertos en Excel), que a su vez contiene objetos **Worksheet** (hojas de cálculo de un libro). El contenedor principal es el objeto **Application**.

Un conjunto de objetos del mismo tipo constituye una **colección** (colección Workbooks: conjunto de libros abiertos en Excel; colección Worksheets: conjunto de hojas de cálculo de un libro).

Un objeto dispone de un conjunto de características llamadas **propiedades** (por ejemplo, para el objeto **Application**: la propiedad **UserName** representa el nombre del usuario, la propiedad **Version** devuelve el número de versión de Microsoft Excel) y de comportamientos o acciones llamados **métodos** (por ejemplo, para el objeto **Application**, el método **FindFile** muestra el cuadro de diálogo **Abrir**; el método **Quit** sale de Excel, etc.).

A un objeto le suceden **eventos** provocados por el usuario (por ejemplo: la apertura de un libro, un clic en un botón de comando, el cambio de la celda activa, etc.) o por el sistema.

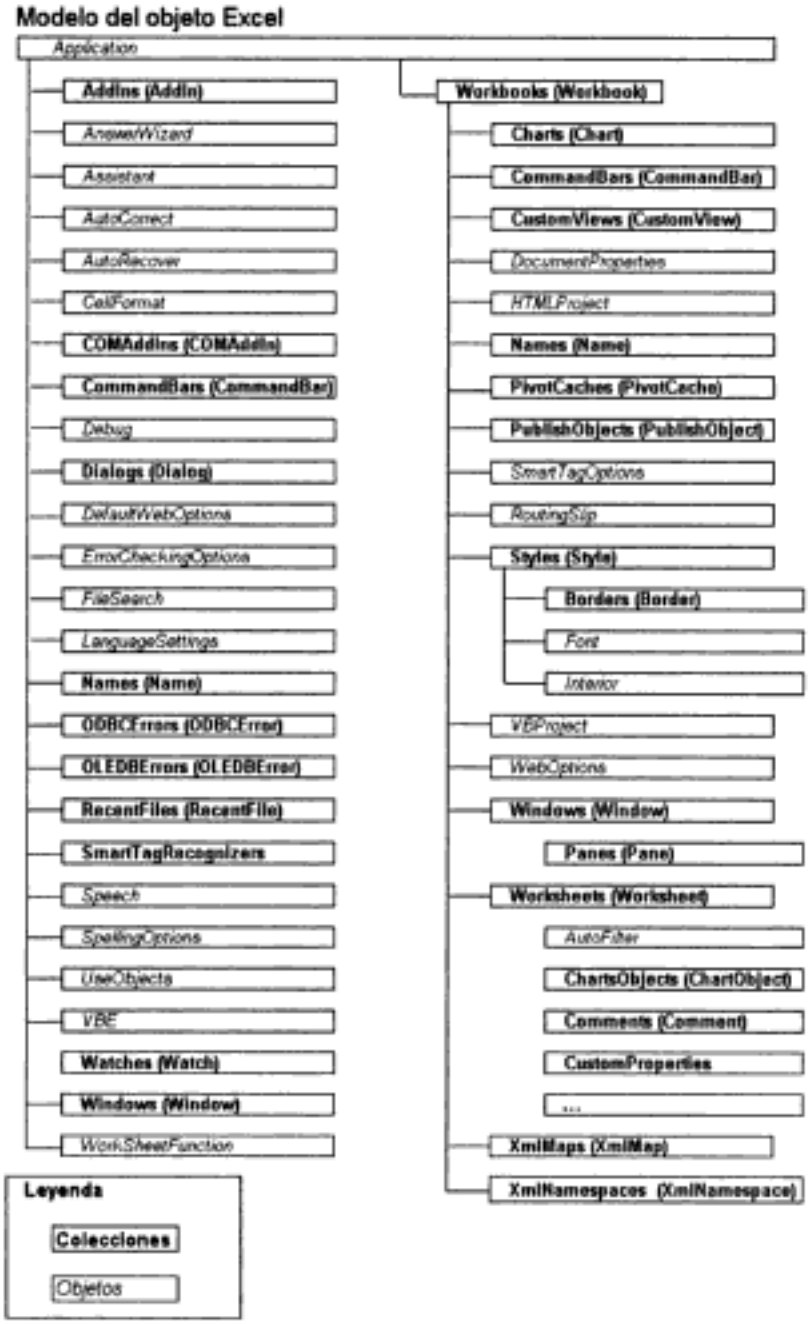
Las **clases** son modelos que permiten crear objetos de un mismo tipo. Los objetos de una misma clase heredan sistemáticamente todos los métodos, propiedades y eventos de su clase. Es posible crear clases de objetos con VBA Excel usando módulos de clase.

Copyrighted Material

B. El modelo de objetos de Excel

1. Presentación

Este cuadro permite distinguir los principales objetos y colecciones de objetos de Excel.



© Editions ENI - All rights reserved

Copyrighted material

2. Objetos y colecciones

La siguiente lista resume la utilidad de los objetos y colecciones del modelo.

Objetos

Application

Objeto que hace referencia a la aplicación Microsoft Excel activa.

AnswerWizard

Objeto que representa la ayuda intuitiva de Excel.

Assistant

Objeto que representa el Ayudante de Office. Usa la propiedad **Visible** para mostrar el Ayudante y la propiedad **On** para activarlo.

AutoCorrect

Objeto que contiene los atributos de Autocorrección de Microsoft Excel.

AutoRecover

Objeto que representa las opciones de recuperación automática de un libro. Estas macros son accesibles desde Excel a partir de la ficha **Guardar** del menú **Herramientas - Opciones**.

CellFormat

Objeto que representa los criterios de búsqueda en el formato de las celdas (se usa con las propiedades **FindFormat** y **ReplaceFormat** del objeto Application).

Debug

Objeto que muestra los datos en la ventana Inmediato, durante la ejecución del código.

DefaultWebOptions

Objeto que contiene los atributos usados por Excel para la apertura o grabación de una página Web.

Copyrighted material

ErrorCheckingOptions

Objeto que contiene las opciones de comprobación de errores de la aplicación Excel.

LanguageSettings

Objeto que contiene información sobre la configuración de idioma de Excel.

Speech

Objeto que contiene los métodos y propiedades que se relacionan con las funciones de síntesis de voz.

SpellingOptions

Objeto que representa las opciones de ortografía de la aplicación.

VBE

Objeto que representa Visual Basic Editor.

WorkSheetFunction

Objeto que contiene todas las funciones disponibles en Excel. Este objeto permite obtener el resultado de una función aplicada a un rango de celdas.
Ejemplo:
Prom = Application.WorksheetFunction.Average(Selection)

Colecciones

AddIns

Colección que contiene todos los complementos (objetos AddIn). A estos complementos se llega desde Excel a través del menú **Herramientas - Complementos**.

COMAddIns

Representa los complementos COM actualmente instalados en Microsoft Excel.

CommandBars

Colección de barras de herramientas de la aplicación activa (objetos CommandBar).

© Editions ENI - All rights reserved

Copyrighted Material

Dialogs

Colección de los cuadros de diálogo integrados de Excel.

Names

Colección de todos los nombres (celdas y rangos con nombre) del libro activo.

ODBCErrors

Colección de todos los errores ODBC generados por la última operación efectuada en un informe de tabla dinámica o en una tabla de consulta.

OLEDBErrors

Colección que representa la información relacionada con el error devuelto por la consulta OLEDB más reciente.

RecentFiles

Colección de los últimos archivos usados.

SmartTagRecognizers

Colección de motores de reconocimientos que devuelven etiquetas inteligentes. Esta lista es accesible desde Excel a través de la ficha **Etiquetas inteligentes** del menú **Herramientas - Opciones de Autocorrección**.

UsedObjects

Colección de los objetos usados en Excel.

Watches

Colección de objetos que representa los rangos de inspección cuando la hoja de cálculo es recalculada.

Windows

Colección de todas las ventanas de la aplicación Excel o de un libro.

Workbooks

Colección de los libros (objeto Workbook) abiertos.

Worksheets

Colección de las hojas de cálculo (objeto Worksheet) de un libro.

Copyrighted material

C. Principios de uso de los objetos y las colecciones

1. Propiedades

Las propiedades sirven para describir un objeto. Ciertas propiedades son de sólo lectura y no pueden, por lo tanto, modificarse con el código VBA.

Sintaxis

```
{<objeto> | <variable objeto>}.<propiedad>
```

Ejemplo

```
'   Modificación del puntero del ratón
Application.Cursor = xlWait

'   Muestra la versión de la aplicación Excel activa
'   Esta propiedad es de sólo lectura
MsgBox Application.Version
Application.Cursor = xlDefault
```

2. Propiedades que representan objetos

Los objetos globales y los objetos definidos en el código a través de clases manejadas por VBA tienen ciertas propiedades cuyo valor se actualiza automáticamente por el sistema.

Estas **propiedades específicas** permiten acceder directamente a ciertos objetos: ventana activa, libro activo, celdas de la hoja activa, etc. La siguiente tabla muestra las propiedades específicas más usadas.

© Editions ENI - All rights reserved

Copyrighted Material

Propiedad	Objeto Parent	Objeto devuelto
ActiveCell	Application Window	Objeto **Range** que representa la primera celda activa de la ventana activa o especificada.
ActiveChart	Application Window Workbook	Objeto **Chart** que representa el gráfico activo.
ActiveControl	Frame Page UserForm	Objeto **Control** que representa el control (ActiveX) activo.
ActiveMenuBar	CommandBars	Objeto **CommandBar** que representa la barra de menú activa en Excel.
ActivePane	Window	Objeto **Pane** que representa el panel activo de la ventana activa.
ActiveSheet	Application Window Workbook	Objeto **Worksheet** que representa la hoja activa del libro activo o del libro especificado.
ActiveWindow	Application	Objeto **Window** que representa la ventana activa.
ActiveWork-Book	Application	Objeto **Workbook** que representa el libro de la ventana activa.
Next Previous	Chart Range Worksheet	Devuelve respectivamente el próximo (Next) o el anterior (Previous) objeto de un mismo tipo (Chart, Range o Worksheet).
Parent	Objetos múltiples	Devuelve el objeto contenedor.
Selection	Application Windows	Objeto **Range** que representa la o las celdas seleccionadas.
ThisCell	Application	Devuelve la celda por la que la función definida por el usuario es llamada como objeto Range.

Copyrighted Material

Propiedad	Objeto Parent	Objeto devuelto
ThisWorkbook	Application	Objeto **Workbook** que representa el libro sobre el que se ejecuta el código de la macro actual.

Las propiedades específicas que devuelven un objeto Range (Cells, Offset, Columns, Rows, etc.) se explican en detalle en el capítulo 4.

3. Métodos

Los métodos permiten realizar acciones sobre los objetos.

Son similares a los procedimientos:

- pueden usar o no argumentos,
- ciertos métodos pueden devolver un valor como los procedimientos **Function**, otros no, como los procedimientos **Sub**.

Sintaxis de método que no devuelve un valor

```
{<objeto> | <variable objeto>}.<método>
[<Lista de argumentos>]
```

Ejemplo

```
'   Activa la segunda hoja de cálculo del libro activo
Application.Goto ActiveWorkbook.Worksheets(2).Range("A1")
'   Selecciona un rango de celdas
Range("A1:C12").Select
'   Borra las celdas seleccionadas
Selection.Clear
'   Guarda el libro activo con un nuevo nombre
ActiveWorkbook.SaveAs "C:\presupuesto\presupuesto2.xls"
```

Como en el caso de los procedimientos, los distintos argumentos de un método se separan por comas. Si un argumento opcional no se define explícitamente, el método usará un valor por defecto.

© Editions ENI - All rights reserved

Copyrighted material

Sintaxis de método que devuelve un valor

```
<variable> = {<objeto> | <variable objeto>}.<método>
[<Lista de argumentos>]
```

Ejemplo

```
'   Muestra el cuadro de diálogo Abrir
Dim strFileName as Variant= Application.GetOpenFilename _
      (FileFilter:="Libros Excel (*.xlsm), *.xlsm", _
      Title:="Seleccione el archivo que desea abrir")
'   Si selecciona un archivo, lo abre
If strFileName <> False Then
   Workbooks.Open strFileName
End If
```

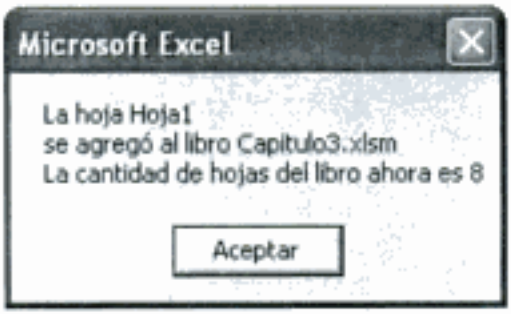
4. Eventos

Un evento es una **acción específica** que se realiza sobre un cierto objeto. Microsoft Excel responde a varios tipos de eventos: apertura o cierre de un libro, selección de celdas, agregar una hoja de cálculo, etc. Los eventos resultan generalmente de una acción del usuario.

El uso de un procedimiento asociado a eventos le permite definir su propio código como respuesta a un evento que se produce en un libro, una hoja o un formulario.

Ejemplo

Cuando se agrega una nueva hoja de cálculo al libro, aparece un mensaje para el usuario.



Copy Grupo Material

```
Private Sub Workbook_NewSheet(ByVal Sh As Object)

MsgBox "La hoja " & Sh.Name & Chr(13) & _
       "se agregó al libro " & _
       ActiveWorkbook.Name & Chr(13) & _
       "La cantidad de hojas del libro ahora es " & _
       ActiveWorkbook.Worksheets.Count

End Sub
```

La administración de eventos es uno de los aspectos más importantes en el desarrollo de aplicaciones Excel; el capítulo 8 está íntegramente dedicado a este tema.

5. Colecciones

Para hacer referencia a un objeto de una colección, puede usarse alguna de las siguientes sintaxis:

```
NomColección!NomObjeto
NomColección![NomObjeto]
NomColección("NomObjeto")
NomColección(var)
```

donde `var` representa una variable de tipo String que contiene el nombre del objeto.

```
NomColección(index)
```

donde `index` representa el número del índice del objeto en la colección.

Para asegurar una mejor legibilidad del código, se aconseja usar siempre la misma sintaxis. Las sintaxis tercera y quinta se recomiendan porque permiten activar el asistente del editor de código. Además, la quinta sintaxis es más útil para recorrer los objetos de una colección.

© Editions ENI - All rights reserved

Copyrighted material

Atención: el primer elemento de la mayoría de las colecciones lleva el índice 1. Use los índices solamente para recorrer una colección. Evite, por ejemplo, usar ActiveWorkbook.ActiveSheet(3) para hacer referencia a una hoja de cálculo del libro activo, ya que el índice de la hoja puede cambiar (si mueve las hojas o si elimina alguna).

Ejemplo

El siguiente código activa la hoja de cálculo Hoja1 del libro Presupuesto.xls. El código usa las colecciones Workbooks y Worksheets.

```
Workbooks("Presupuesto.xls").Worksheets("Hoja1").Activate
```

o

```
Workbooks![Presupuesto.xls].Worksheets!Hoja1.Activate
```

o

```
Workbooks![Presupuesto.xls].Worksheets![Hoja1].Activate
```

Recorrer una colección: este código cambia el nombre de las hojas de cálculo del libro activo.

```
Dim i As Integer
For i = 1 To ActiveWorkbook.Worksheets.Count
      ActiveWorkbook.Worksheets(i).Name = "Presupuesto Nº " & i
Next i
```

Una colección también puede recorrerse con la instrucción For Each Next.

```
Dim Hoja As Worksheet
For Each Hoja In ActiveWorkbook.Worksheets
      Hoja.Name = "Presupuesto Nº " & Hoja.Index
Next Hoja
```

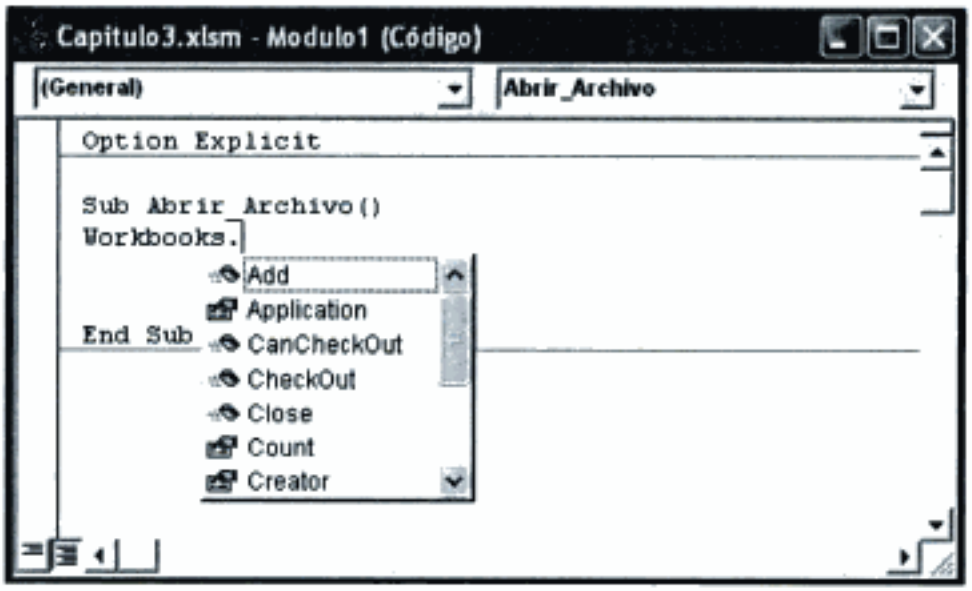

Copyrighted Material

6. Redacción automática de instrucciones

El editor VBA incluye una tecnología que lo ayuda en el empleo de objetos. Cuando usted escribe el nombre de un objeto o de una colección reconocida por VBA seguido de un punto, se despliega una lista con los métodos y las propiedades del objeto. Si selecciona un método, el asistente lo ayuda a indicar los argumentos que le corresponden.

Ejemplo

→ Escriba el nombre de la colección Workbooks seguido de un punto, aparece la siguiente lista desplegable.



El icono representa los métodos, el icono representa las propiedades y las colecciones.

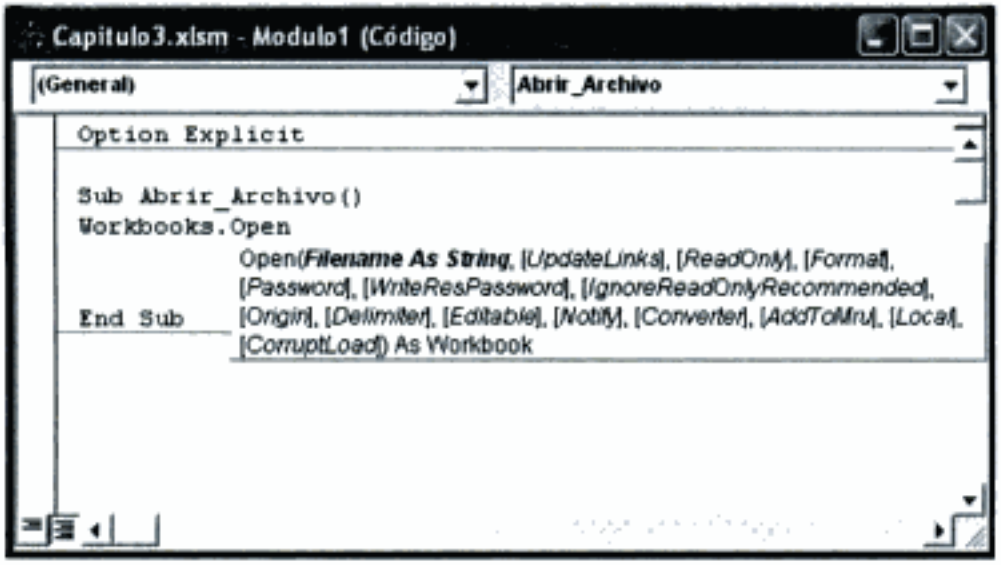
→ Usted puede hacer avanzar la lista escribiendo las primeras letras del método, propiedad o colección buscada o con la ayuda de la barra de desplazamiento. Para seleccionar un elemento de la lista, haga doble clic sobre él.

→ Escriba un punto si acaba de seleccionar un objeto para ver la lista de sus propiedades y métodos. Si selecciona un método, escriba un espacio para obtener la lista de configuración del método.

→ Para seguir este ejemplo, seleccione el método **Open**, y luego escriba un espacio.

© Editions ENI - All rights reserved

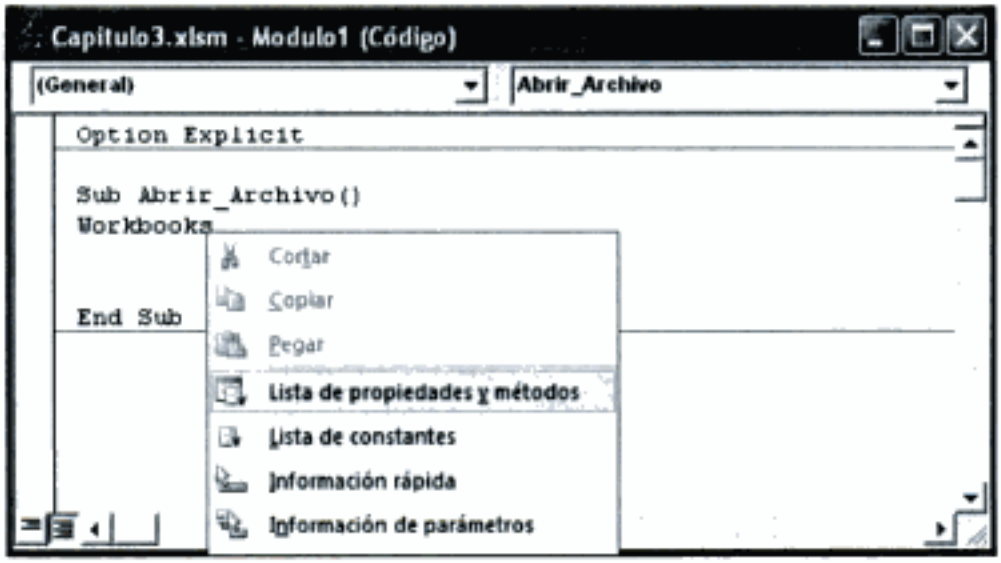
Copyrighted material



Aparecerá una lista con los argumentos del método. Los argumentos opcionales aparecen entre corchetes. El argumento actual se verá en negrita. Si para un argumento dado existe una lista de valores predefinidos, aparecerá una lista desplegable con las constantes correspondientes.

También puede activar la lista de propiedades y métodos de la siguiente manera:

- Ubique el cursor detrás del punto situado después del método.
- Haga un clic derecho para hacer aparecer el menú contextual.
- Seleccione la opción **Lista de propiedades y métodos**.



Copyrighted material

D. Instrucciones usadas con los objetos

1. La instrucción With

La instrucción **With** permite acceder varias veces al mismo objeto, indicándolo una sola vez.

Ofrece varias ventajas:

- optimización del tiempo de ejecución del código,
- ganar tiempo en la escritura del código,
- mejor legibilidad del código.

Sintaxis

```
With <Objeto>
    <código que usa los métodos y propiedades>
    <que se relacionan con el objeto>
End With
```

Ejemplo

Agregar y modificar una hoja de cálculo.

```
With ActiveWorkbook
   '   Crea una hoja después de la última hoja del libro activo
   .Worksheets.Add , .Worksheets(.Worksheets.Count)
   '   Modifica el nombre de la nueva hoja y el valor de la celda A1
   With .ActiveSheet
      .Name = "Síntesis"
      .Range("A1") = "Revisión del Presupuesto"
   End With
End With
```

2. La instrucción For Each...Next

La instrucción **For Each... Next** permite recorrer los objetos de una colección o de una tabla.

© Editions ENI - All rights reserved

Copyrighted Material

Sintaxis

```
For Each <elemento > In <grupo>
    <secuencia de instrucciones>
    [Exit For]
    <secuencia de instrucciones>
Next <elemento>
```

Ejemplo

Modificación del contenido de la celda A1 y de la cantidad de hojas del libro activo.

```
Sub NumPresupuesto()
   Dim Hoja As Worksheet

   For Each Hoja In ActiveWorkbook.Worksheets
         Hoja.Cells(1, 1) = "PRESUPUESTO N° " & Hoja.Index
      Hoja.Name = "Presupuesto " & Hoja.Index
   Next Hoja
End Sub
```

3. La instrucción If TypeOf

La instrucción **If TypeOf** permite comprobar el tipo de un objeto.

Sintaxis

```
If TypeOf <Objeto> Is <TipoObjeto> Then
    <código que usa los métodos y propiedades>
    <relacionados con el objeto>
End If
```

Ejemplo

```
If TypeOf obj.Parent Is Worksheet Then_...
```


Copyrighted Material

4. La instrucción Set

La instrucción **Set** permite atribuir la referencia de un objeto a una variable, llamada **variable objeto**.

Esta instrucción puede usarse para **crear un nuevo objeto** (usando en ese caso un método que permita crear el objeto) o para **hacer referencia a un objeto** ya existente.

Sintaxis

```
Set <Objeto> = [New] <expresión objeto>
O Set <Objeto> = Nothing
```

`objeto`	es una variable de tipo String, que contiene el nombre del objeto que hay que crear.
La palabra clave `New`	permite crear una nueva instancia de la clase. Si la variable <Objeto> contiene una referencia a un objeto, esta última es abandonada.
`<expresión objeto>`	puede ser el nombre de un objeto o de una variable objeto del mismo tipo, o sea, una función o método que devuelva un objeto del mismo tipo.
`Nothing`	reinicializa la variable objeto y libera el conjunto de recursos del sistema y la memoria asociadas al objeto.

Ejemplo

Creación de un libro con dos hojas y asignación de un nombre a cada una de ellas.

```
Dim Libro As Workbook
Dim i As Integer

'   Creación de un nuevo libro
Set Libro = Application.Workbooks.Add
'   Eliminar las hojas a partir de la tercera
With Libro
   For i = 3 To .Worksheets.Count                    .../...
```

© Editions ENI - All rights reserved

Copyrighted material

```
.../...
        .Worksheets(i).Delete

   Next i
'   Asignar los nombres a las hojas 1 y 2
   .Worksheets(1).Name = "Ventas Año 2006"
   .Worksheets(2).Name = "Ventas Año 2007"
   .SaveAs "C:\Ventas\Histórico"
End With
```

Modificar una hoja en un libro abierto.

```
Dim Libro As Workbook
Dim Hoja As Worksheet

Set Libro = Application.Workbooks![Histórico.xls]
Set Hoja = Libro.Worksheets![Ventas Año 2007]
With Hoja
      .Name = "Ventas 2007"
      .Range("A2") = "Ventas del año 2007"
End With
Set Hoja = Nothing
```

E. El Examinador de objetos

1. Presentación

Dada la cantidad y diversidad de objetos Excel, es útil poder encontrar la información que se relaciona con los distintos objetos.

El Examinador de objetos muestra la información relativa a los objetos, métodos, propiedades, eventos y constantes.

Usted puede obtener el Examinador de objetos de distintas maneras:

→) **Ver**
Examinador de objetos [F2]
(barra de herramientas **Estándar**)

Copyrighted Material

1. **Lista de bibliotecas** actualmente cargada.

2. **Texto buscado**: objeto, propiedad, colección, evento, método, etc.

3. **Resultado de la búsqueda**: lista de las clases de objetos (objetos y colecciones) y de sus miembros (objeto, colección, propiedad, evento o método). La palabra buscada puede estar en la lista de clases o en la de sus miembros.

4. **Clases de objetos de la biblioteca**: la clase de objeto seleccionada en la lista **Resultado** aparece remarcada.

5. **Propiedades** (icono), **métodos**, (icono), **eventos** (icono) y **constantes** (icono) se relacionan con la clase de objeto seleccionada o remarcada en la lista de la izquierda.

6. **Detalle** del elemento seleccionado.

© Editions ENI - All rights reserved

Copyrighted material

2. Búsqueda en el Examinador de objetos

Para hacer una búsqueda en el Examinador de objetos, proceda de la siguiente manera:

- Índique la palabra buscada en la segunda lista desplegable.
- Haga clic en el icono . Si la ventana **Resultados** muestra muchas líneas, desplácese hasta la que le interese; la parte inferior de la ventana se actualizará.

Copyrighted Material

Capítulo 4: Objetos de Excel

A. El objeto Application 109

1. Propiedades que representan las opciones de Excel 109

a. Opciones de la categoría Más frecuentes 110

b. Opciones de la categoría Fórmulas 111

c. Opciones de la categoría Revisión 113

d. Opciones de la categoría Guardar 114

e. Opciones de la categoría Avanzadas 115

2. Propiedades relativas a la presentación de la aplicación . . 119

3. Propiedades diversas 120

4. Métodos del objeto Application 123

a. Métodos que actúan sobre las fórmulas y cálculos . . . 123

b. Métodos que actúan sobre las celdas 124

c. Métodos que actúan sobre las listas personalizadas . . 124

d. Métodos que muestran los cuadros de diálogo 125

e. Métodos relacionados con las acciones en Excel 125

f. Métodos relativos al correo 126

g. Métodos relativos al protocolo DDE 126

h. Otros métodos 127

Copyrighted material

5. Ejemplos de códigos que usan el objeto Application 128
a. Modificación de la interfaz de Excel 128
b. Creación de una lista personalizada 128
c. Selección de columnas no consecutivas 129
d. Evaluación del resultado de una fórmula 129

B. Objeto Workbook 130

1. Objetos y colecciones 131
2. Propiedades 133
a. Propiedades relativas a la actualización y grabado de libros 133
b. Propiedades relativas a libros compartidos 134
c. Otras propiedades 136
3. Lista de métodos 138
a. Métodos que actúan directamente sobre los libros . . . 138
b. Métodos relativos a la seguridad 140
c. Métodos relativos a libros compartidos 140
d. Métodos relacionados con datos vinculados 141
e. Métodos relativos al envío de libros 142
f. Otros métodos 143
4. Ejemplos de códigos que usan el objeto Workbook 143
a. Creación de un libro Excel 143
b. Importar una base de datos y exportarla al formato HTML 144
c. Mostrar las propiedades de un libro 145
d. Exportar un libro al formato pdf 145

Copyrighted material

C. El objeto Worksheet 146

1. Lista de objetos y colecciones 146
2. Objetos y colecciones 147
3. Propiedades 149
4. Métodos 151
5. Ejemplos de códigos que usan el objeto Worksheet 153
   a. Ordenar las hojas de un libro 153
   b. Protección de las hojas de cálculo de un libro 155
   c. Ordenar una tabla 155

D. El objeto Range 156

1. Propiedades y métodos que devuelven un objeto Range . . 156
2. Sintaxis de las propiedades que devuelven un objeto Range 158
3. Lista de objetos y colecciones 163
4. Propiedades 165
   a. Propiedades relacionadas con la posición y el formato de las celdas 165
   b. Propiedades que se relacionan con el contenido de celdas y con las fórmulas 166
   c. Otras propiedades 167

Copyrighted Material

5. Métodos 167

a. Métodos que devuelven un objeto 167

b. Métodos que se relacionan con la presentación de las celdas 168

c. Métodos que se relacionan con el contenido de las celdas 169

d. Métodos que se relacionan con los nombres de celdas . 171

e. Métodos que se relacionan con los filtros 171

f. Métodos que se relacionan con el modo esquema . . . 172

g. Métodos que se relacionan con la herramienta de Auditoría 172

h. Otros métodos 173

E. Ejemplos de uso de los objetos 174

1. Cálculo del importe de un premio 174

2. Asignación de comentarios a las celdas 176

Copyrighted material

A. El objeto Application

El objeto **Application** representa la aplicación Microsoft Excel activa. Es el objeto por defecto y, por lo tanto, es generalmente opcional (ejemplo: *Version* equivale a *Application.Version*).

Este objeto contiene:

- las **propiedades relativas al entorno Excel** (opciones de Excel, etc.) y a la **presentación de la interfaz** (puntero del ratón, texto de la barra de estado, tamaño y estado de la ventana de la aplicación, etc.),
- hay distintos **métodos** para realizar acciones en el entorno Excel,
- **propiedades** que devuelven objetos y colecciones de primer nivel (objetos y colecciones del modelo de objetos Excel: Workbooks, Charts, etc.),
- **propiedades específicas** que hacen referencia directa a objetos: ActiveCell, ActiveSheet, ActiveWindow, etc.

Las propiedades que hacen referencia a objetos se explican en el capítulo anterior.

1. Propiedades que representan las opciones de Excel

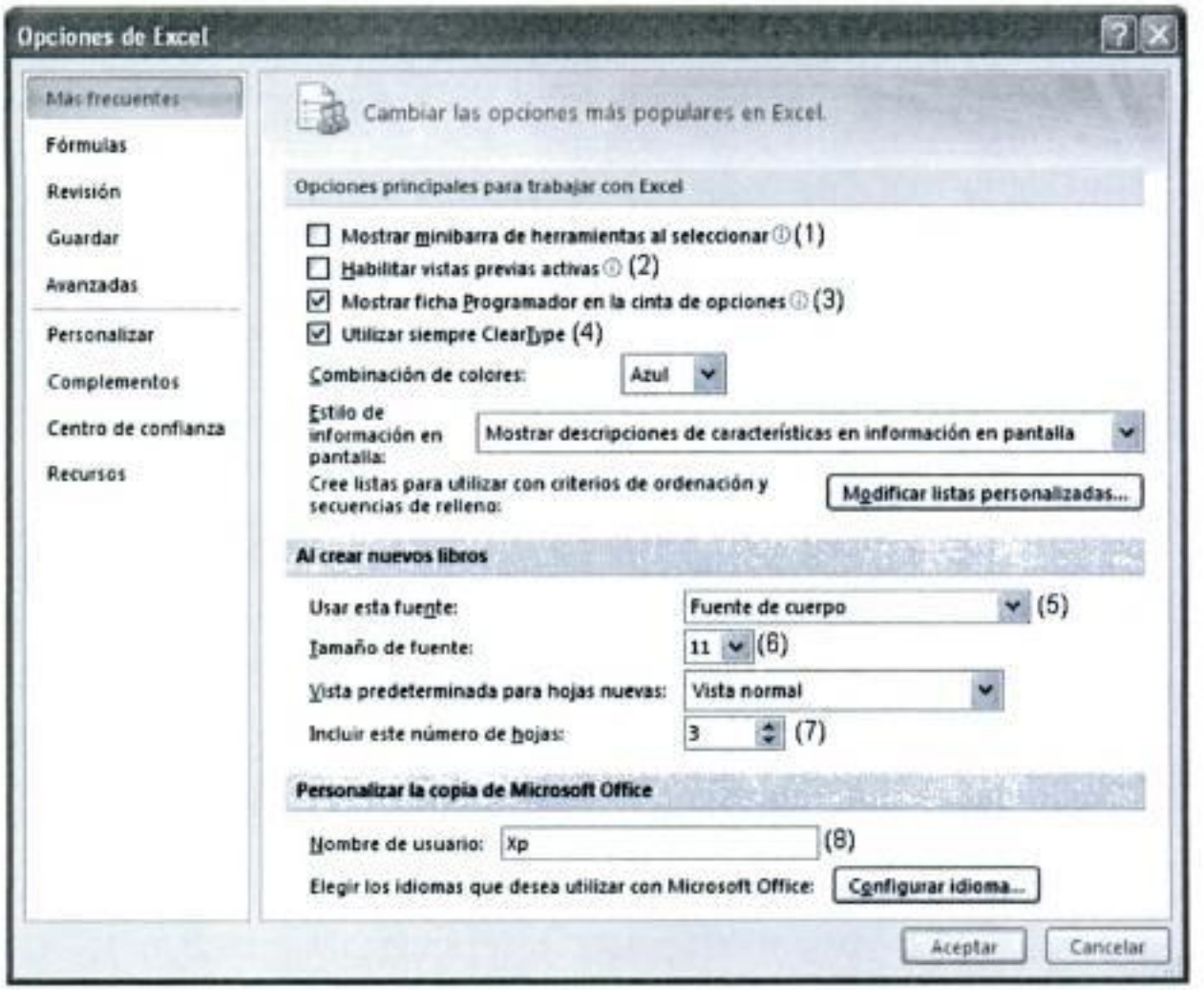
Las principales opciones de Excel pueden ser devueltas o definidas a partir de propiedades del objeto **Application**. La mayoría de estas propiedades son de lectura y escritura.

Para acceder a las opciones Excel en la versión 2007, haga clic en el botón **Microsoft Office** y luego en el botón **Opciones de Excel** (situado en la parte inferior del cuadro de diálogo).

© Editions ENI - All rights reserved

Copyrighted material

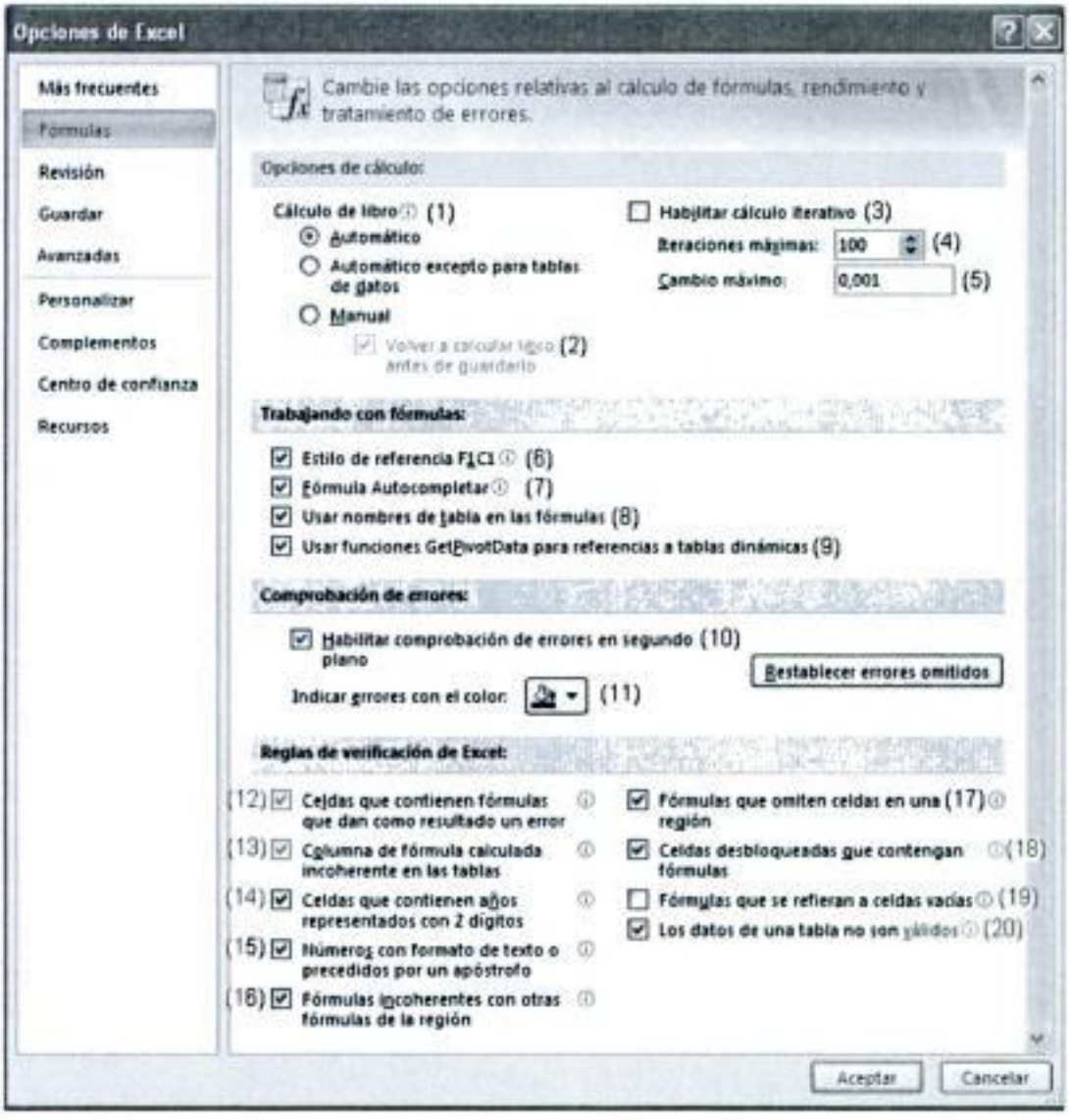
a. Opciones de la categoría Más frecuentes



N°	Propiedades	Valores devueltos
1	ShowSelectionFloaties	Boolean
2	EnableLivePreview	Boolean
3	ShowDevTools	Boolean
4	AlwaysUseClearType	Boolean
5	StandardFont.	Entero largo
6	StandardFontSize	Entero largo
7	SheetsInNewWorkbook	Entero largo
8	UserName	Cadena de caracteres

Copyrighted material

b. Opciones de la categoría Fórmulas



N°	Propiedades	Valores devueltos
1	Calculation	Constante
	Constantes:	**xlCalculationAutomatic**
		xlCalculationManual
		xlCalculationSemiautomatic
2	CalculateBeforeSave	Boolean
3	Iteration	Boolean
4	MaxIterations	Entero largo
5	MaxChange	Doble

© Editions ENI - All rights reserved

Copyrighted Material

N°	Propiedades	Valores devueltos
6	ReferenceStyle	
		Constantes: **xlA1** **xlR1C1**
7	DisplayFormulaAutoComplete	Boolean
8	GenerateTableRefs	Constantes: **xlGenerateTableRefStruct** **xlGenerateTableRefA1**
9	GenerateGetPivotData	Boolean

Opciones de comprobación de errores

Las siguientes propiedades dependen de la propiedad **ErrorCheckingOptions** del objeto Application. Esta propiedad devuelve un objeto **ErrorCheckingOptions**, que representa las opciones de comprobación de errores para una aplicación.

N°	Propiedades	Valores devueltos
10	BackgroundChecking	Boolean
11	IndicatorColorIndex	Constante XIColorIndex
12	EvaluateToError	Boolean
13	InconsistentTableFormula	Boolean
14	TextDate	Boolean
15	NumberAsText	Boolean
16	InconsistentFormula	Boolean
17	OmittedCells	Boolean
18	UnlockedFormulaCells	Boolean
19	EmptyCellReferences	Boolean
20	ListDataValidation	Boolean

Copyrighted material

c. Opciones de la categoría Revisión

Opciones de revisión ortográfica

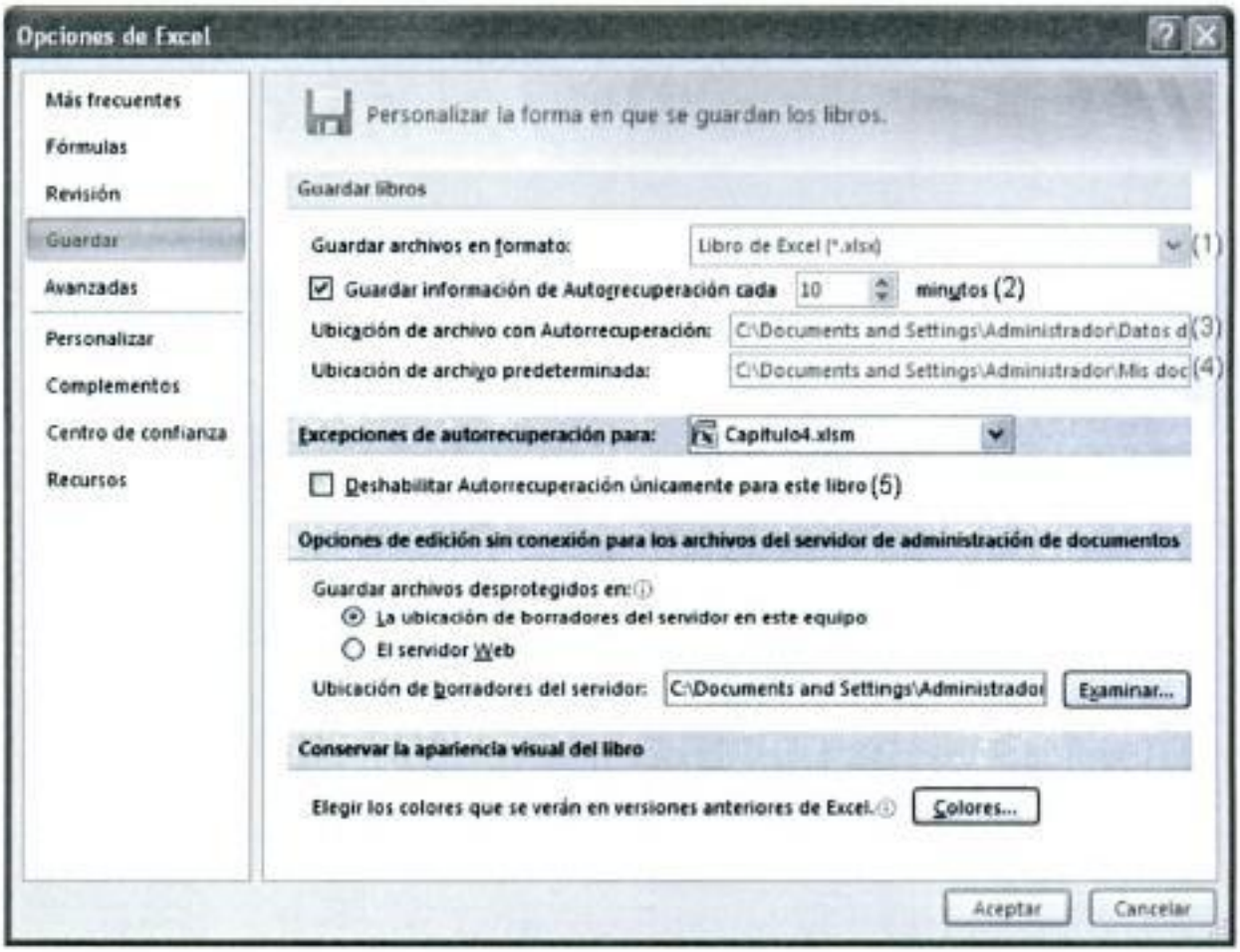
Las siguientes propiedades dependen de la propiedad **SpellingOptions** del objeto **Application**. Esta propiedad devuelve un objeto **SpellingOptions**, que representa las opciones de revisión ortográfica para una aplicación.

N°	Propiedades	Valores devueltos
1	IgnoreCaps	Boolean
2	IgnoreMixedDigits	Boolean
3	IgnoreFileNames	Boolean
4	GermanPostReform	Boolean
5	SuggestMainOnly	Boolean
6	ArabicModes	Constante XlArabicModes
7	DictLang	Entero largo

© Editions ENI - All rights reserved

Copyrighted material

d. Opciones de la categoría Guardar

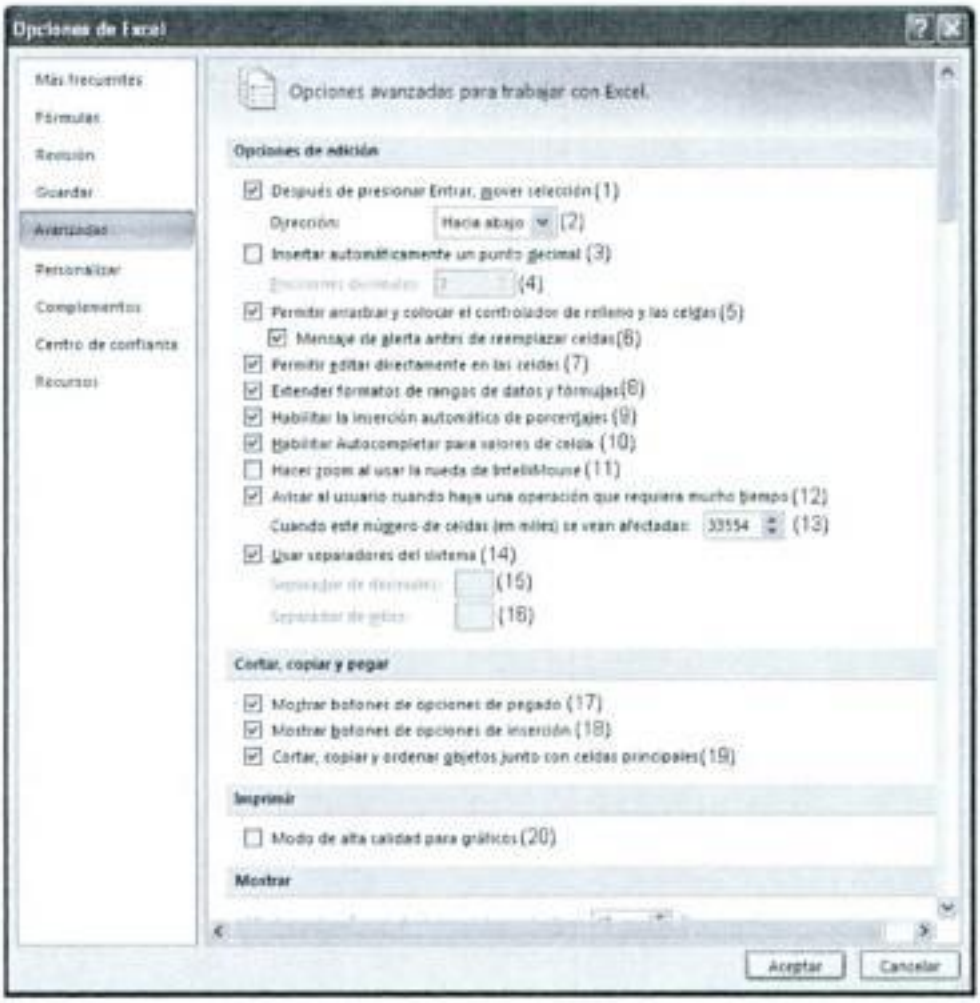


N°	Propiedades	Valores devueltos
1	DefaultSaveFormat	Constante xlFileFormat
2	AutoRecover.Time	Entero largo
3	AutoRecover.Path	Cadena de caracteres
4	DefaultFilePath	Cadena de caracteres
5	EnableAutoRecover	Boolean

La opción 5 se aplica al libro activo (Application.ActiveWorkbook).

Copyrighted Material

e. Opciones de la categoría Avanzadas

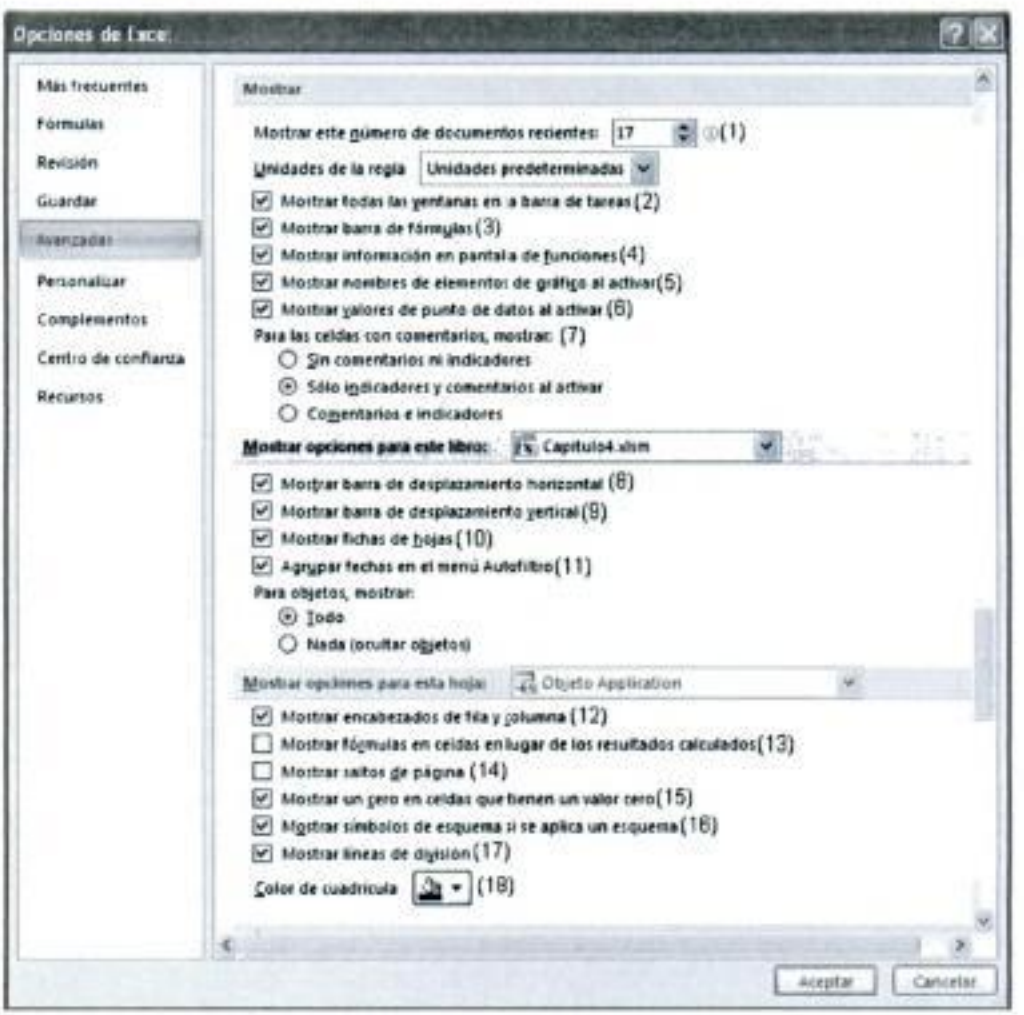


N°	Propiedades	Valores devueltos
1	MoveAfterReturn	Boolean
2	MoveAfterReturnDirection	Constante
	Constantes:	**xlDown**
		xlUp
		xltoRight
		xltoLeft
3	FixedDecimal	Boolean
4	FixedDecimalPlaces	Entero largo
5	CellDragAndDrop	Boolean
6	AlertBeforeOverwriting	Boolean
7	EditDirectlyInCell	Boolean

© Editions ENI - All rights reserved

Copyrighted material

N°	Propiedades	Valores devueltos
8	ExtendList	Boolean
9	AutoPercentEntry	Boolean
10	EnableAutoComplete	Boolean
11	RollZoom	Boolean
12	EnableLargeOperationAlert	Boolean
13	LargeOperationCellThousandCount	Entero largo
14	UseSystemSeparators	Boolean
15	DecimalSeparator	Cadena de caracteres
16	ThousandsSeparator	Cadena de caracteres
17	DisplayPasteOptions	Boolean
18	DisplayInsertOptions	Boolean
19	CopyObjectsWithCells	Boolean
20	HighQualityModeForGraphics	Boolean



Copyrighted Material

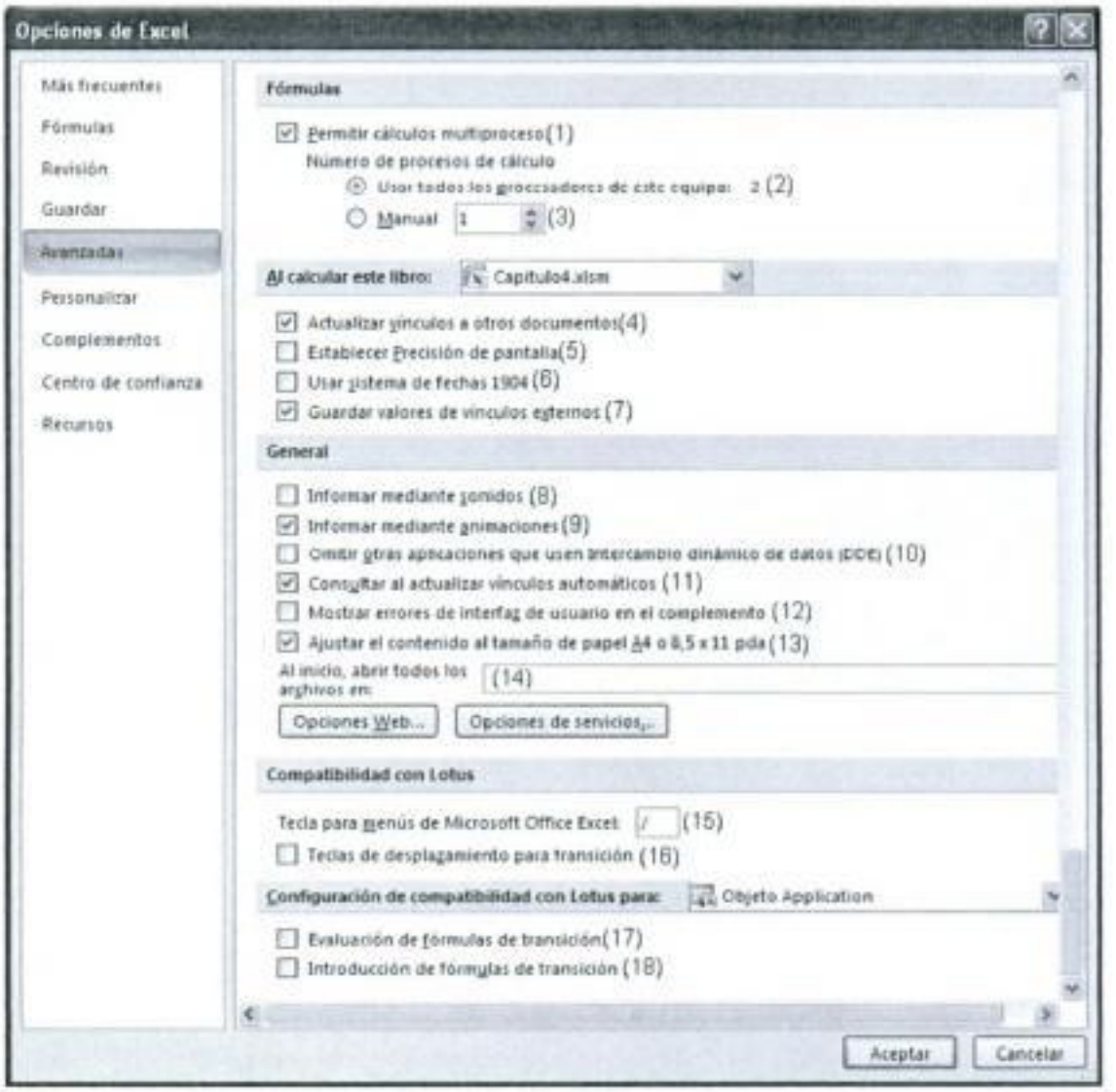
N°	Propiedades	Valores devueltos
1	RecentFiles.Maximum	Entero largo
2	DisplayFormulaBar	Boolean
3	ShowWindowsInTaskbar	Boolean
4	DisplayFunctionToolTips	Boolean
5	ShowChartTipNames	Boolean
6	ShowChartTipValues	Boolean
7	DisplayCommentIndicator	Constante
	Constantes:	**xlNoIndicator** **xlIndicatorOnly** **xlCommentAndIndicator**

Las siguientes propiedades dependen de la propiedad **ActiveWindow** del objeto **Application**. Esta propiedad devuelve un objeto **Window** que representa la ventana activa.

N°	Propiedades	Valores devueltos
8	DisplayHorizontalScrollBar	Boolean
9	DisplayVerticalScrollBar	Boolean
10	DisplayWorkbookTabs	Boolean
11	AutoFilterDateGrouping	Boolean
12	DisplayHeadings	Boolean
13	DisplayFormulas	Boolean
14	DisplayPageBreaks	Boolean
15	DisplayZeros	Boolean
16	DisplayOutline	Boolean
17	DisplayGridlines	Boolean
18	GridlineColorIndex	Constante XlColorIndex

© Editions ENI - All rights reserved

Copyrighted Material



N°	Propiedades	Valores devueltos
1	MultiThreadedCalculation.Enabled	Boolean
2	MultiThreadedCalculation.ThreadMode Constantes:	Constante **xlThreadModeAutomatic** **xlThreadModeManual**
3	MultiThreadedCalculation.ThreadCount	Entero
4	SaveLinkValues	Boolean
5	PrecisionAsDisplayed	Boolean
6	Date1904	Boolean
7	UpdateRemoteReferences	Boolean
8	EnableSound	Boolean

Copyrighted Material

N°	Propiedades	Valores devueltos
9	EnableAnimations	Boolean
10	IgnoreRemoteRequests	Boolean
11	AskToUpdateLinks	Boolean
12	WarnOnFunctionNameConflict	Boolean
13	MapPaperSize	Boolean
14	AltStartupPath	Cadena de caracteres
15	TransitionMenuKey	Cadena de caracteres
16	TransitionNavigKeys	Boolean
17	TransitionExpEval	Boolean
18	TransitionFormEntry	Boolean

Las opciones 4 a 7 se aplican al libro activo (Application.ActiveWorkbook) y las opciones 17 y 18, a la hoja activa (Application.ActiveSheet).

2. Propiedades relativas a la presentación de la aplicación

Height

Real doble. Altura de la ventana.

Left

Real doble. Distancia entre el borde izquierdo de la pantalla y el borde izquierdo de la ventana principal de Microsoft Excel.

Top

Real doble. Distancia entre el borde superior de la pantalla y el borde superior de la ventana principal de Microsoft Excel.

Width

Real doble. Distancia entre los bordes izquierdo y derecho de la ventana de la aplicación.

Caption

Cadena de caracteres. Nombre mostrado en la barra de títulos de la ventana Microsoft Excel.

© Editions ENI - All rights reserved

Copyrighted material

Cursor

Constante. Aspecto del puntero del ratón en Excel.

Constantes **xlDefault**: Puntero por defecto
xlIBeam: Puntero en I
xlNorthwestArrow: Flecha Noroeste
xlWait: Reloj de arena

DisplayFullScreen

Boolean. Indica si Excel funciona en modo de pantalla completa.

FormulaBarHeight

Entero largo. Altura, en número de líneas, de la barra de fórmulas.

ShowMenuFloaties

Boolean. Indica si las minibarras de herramientas deben aparecer cuando el usuario hace un clic derecho en la ventana del libro.

StatusBar

Cadena de caracteres. Texto de la barra de estado.

Visible

Boolean. Indica si la ventana principal de la aplicación está visible.

3. Propiedades diversas

AutoFormatAsYouTypeReplaceHyperlinks

Boolean. Indica si Excel convierte en hipervínculos las rutas de Internet de forma automática a medida que se las escribe.

AutomationSecurity

Constante. Modo de seguridad que emplea Microsoft Excel al abrir archivos con macros.

Constantes **msoAutomationSecurityByUI**
msoAutomationSecurityForceDisable
msoAutomationSecurityLow

Copyrighted material

ActivePrinter

Cadena de caracteres. Nombre de la impresora activa.

CalculationInterruptKey

Constante. Indica la tecla que puede interrumpir el recálculo en Microsoft Excel.

Constantes	**xlAnyKey**
	xlEscKey
	xlNoKey

CalculationState

Constante. Indica el estado de cálculo de la aplicación, para el recálculo en curso en Microsoft Excel.

Constantes	**xlCalculating**
	xlDone
	xlPending

ClipboardFormats

Variant. Devuelve una matriz que contiene los formatos que se encuentran en el portapapeles.

CutCopyMode

Constante. Devuelve o define el estado del modo Cortar o Copiar.

Constantes	**False**	No está en modo Cortar ni Copiar
	xlCopy	Está en modo Copiar
	xlCut	Está en modo Pegar

DataEntryMode

Constante. Devuelve o define el modo de entrada de datos.

Constantes	**xlOn**	Modo de entrada de datos activado
	xlOff	Modo de entrada de datos desactivado
	xlStrict	Modo de entrada de datos activado y tecla [Escape] desactivada.

© Editions ENI - All rights reserved

Copyrighted Material

DisplayAlerts

Boolean. Muestra (True) u oculta los mensajes de alerta cuando se ejecuta una macro.

EnableCancelKey

Constante. Controla la forma en que Microsoft Excel trata la manera en que el usuario interrumpe el procedimiento en curso pulsando la combinación de teclas [Ctrl][Pausa].

Constantes	**xlDisabled**	no se interrumpe
	xlInterrupt	interrumpe el procedimiento en curso y pasa al modo Depurar
	xlErrorHandler	en caso de interrupción, el procedimiento genera un error (código de error 18)

EnableEvents

Boolean. Permite desactivar (False) los eventos del objeto Application.

FindFormat

Devuelve o define los criterios de búsqueda para el tipo de formato de celda que hay que encontrar.

GenerateGetPivotData

Boolean. Indica si Excel puede obtener datos de un informe de tabla dinámica.

Hinstance

Entero largo. Devuelve el controlador de instancia de la instancia que llama a Microsoft Excel.

Hwnd

Entero largo. Devuelve un objeto que designa el identificador de ventana superior de la ventana Microsoft Excel.

MapPaperSize

Boolean. Indica si los documentos se ajustan automáticamente cuando cambia el tamaño de papel.

Copyrighted Material

MouseAvailable

Boolean. Indica si hay un ratón disponible.

OrganizationName

Cadena de caracteres. Nombre de la empresa.

PreviousSelections

Variant. Devuelve una matriz de objetos Range que contiene los últimos cuatro rangos seleccionados.

RecordRelative

Boolean. True si las macros se graban usando referencias relativas.

ReplaceFormat

Boolean. Establece los criterios de reemplazo que se emplean para sustituir formatos de celda (se usa con la propiedad **FindFormat**).

TemplatesPath

Cadena de caracteres. Ruta de acceso local de la carpeta donde están guardadas las plantillas.

Version

Cadena de caracteres. Número de versión de la aplicación Excel activa.

4. Métodos del objeto Application

a. Métodos que actúan sobre las fórmulas y cálculos

Calculate

Fuerza un cálculo de los datos para todos los libros abiertos.

CalculateFull

Fuerza un recálculo completo de los datos en todos los libros abiertos.

CalculateFullRebuild

Para todos los libros abiertos, fuerza un recálculo completo de los datos y vuelve a establecer las dependencias.

© Editions ENI - All rights reserved

Copyrighted material

CheckAbort

Detiene el recálculo.

ConvertFormula

Convierte las referencias de celda en una fórmula pasando del estilo de referencia A1 al estilo R1C1.

Evaluate

Calcula la expresión pasada como argumento y devuelve el resultado. La expresión debe corresponder a una fórmula en inglés.

b. Métodos que actúan sobre las celdas

DoubleClick

Equivale a hacer doble clic sobre la celda activa.

GoTo

Selecciona un rango o un procedimiento Visual Basic en cualquier libro y activa ese mismo libro si no lo está.

Intersect

Devuelve un objeto **Range** que representa la intersección de dos o más rangos.

Union

Devuelve la unión de al menos dos rangos.

c. Métodos que actúan sobre las listas personalizadas

AddCustomList

Agrega una lista personalizada.

DeleteCustomList

Elimina una lista personalizada.

GetCustomListContents

Devuelve una lista personalizada (matriz de cadena de caracteres).

Copyrighted material

GetCustomListNum

Devuelve el número de la lista personalizada correspondiente a una matriz de cadena de caracteres.

d. Métodos que muestran los cuadros de diálogo

Los métodos **GetOpenFileName**, **GetSaveAsFileName**, **FindFile** y **InputBox** se explican en el capítulo 5, titulado "Los cuadros de diálogo".

e. Métodos relacionados con las acciones en Excel

ExecuteExcel4Macro

Ejecuta una función macro Microsoft Excel 4.0, y devuelve su resultado.

OnKey

Ejecuta un procedimiento especificado cuando el usuario pulsa una tecla o una combinación de teclas.

OnRepeat

Define el comando del menú **Repetir** y el nombre del procedimiento ejecutado al seleccionar la opción **Repetir** (menú **Edición**) después de la ejecución del procedimiento que define esta propiedad.

OnTime

Programa la ejecución de un procedimiento en un momento determinado.

OnUndo

Define el texto de la opción de menú **Deshacer** y el nombre del procedimiento ejecutado cuando se selecciona la opción **Deshacer** (menú **Edición**) después de la ejecución del procedimiento que define esta propiedad.

Quit

Sale de Microsoft Excel.

Repeat

Repite la última operación ejecutada desde la interfaz de usuario.

© Editions ENI - All rights reserved

Copyrighted Material

SaveWorkspace

Guarda el área de trabajo en curso.

SendKeys

Simula la pulsación de teclas en la aplicación activa.

Undo

Deshace la última operación realizada desde la interfaz de usuario.

RecordMacro

Graba el código si se activa el grabador de macros.

Run

Ejecuta un procedimiento o llama a una función.

Wait

Hace una pausa en la ejecución de la macro hasta un momento especificado. Devuelve el valor **True** cuando llega la hora especificada.

f. Métodos relativos al correo

MailLogon

Se conecta y abre una sesión de correo MAPI o de Microsoft Exchange. Si Microsoft Mail no se está ejecutando aún, se debe usar este método para establecer una sesión de correo para permitir el uso de funciones de distribución de documentos o de mensajes.

MailLogoff

Cierra una sesión de correo MAPI abierta por Microsoft Excel.

g. Métodos relativos al protocolo DDE

Estos métodos (**DDEExecute**, **DDEInitiate**, etc.) se explican en el capítulo 10 - Vínculos entre aplicaciones.

Copyrighted Material

h. Otros métodos

ActivateMicrosoftApp

Activa una aplicación Microsoft. Si ésta ya está en ejecución, el método la activa. Si no lo está, el método abre una nueva instancia de la aplicación.

CentimetersToPoints

Convierte centímetros en puntos (un punto equivale a 0,035 centímetros).

CheckingSpelling

Comprueba la ortografía de una palabra y devuelve **True** si la palabra se encontró en uno de los diccionarios.

DisplayXMLSourcePane

Abre el panel de tareas Office XML Source y muestra la asignación XML especificada por el argumento XmlMap.

Help

Muestra un tema de ayuda.

InchesToPoints

Convierte pulgadas en puntos.

MacroOptions

Opciones del cuadro de diálogo **Opciones de macro**.

RegisterXLL

Carga un recurso de código XLL y registra automáticamente las funciones y comandos que contiene.

Volatile

Define una función personalizada como volátil. Una función volátil se recalcula cada vez que se realiza un cálculo en cualquier celda de la hoja de cálculo.

© Editions ENI - All rights reserved

Copyrighted Material

5. Ejemplos de códigos que usan el objeto Application

a. Modificación de la interfaz de Excel

```
Sub Interfaz()

   With Application
      '   Título de la ventana de la aplicación
      .Caption = "Aplicación " & .Name _
                & " Versión " & .Version
      '   Texto de la barra de estado
      .StatusBar = "Ejemplos VBA Excel 2007"
      '   Estilo de referencia L1C1
     .ReferenceStyle = xlR1C1
      '   Maximiza la ventana de la aplicación
      .WindowState = xlMaximized
      '   Modifica la fuente por defecto
      .StandardFont = "Verdana"
      .StandardFontSize = 11
      '   Oculta la barra de herramientas formato
      '   Empleo de la colección CommandBars
      .CommandBars("Formatting").Visible = False
      '   Muestra la barra de herramientas Visual Basic
      .CommandBars("Visual Basic").Visible = True
   End With

End Sub
```

b. Creación de una lista personalizada

```
Dim i As Integer
Dim NumList As Integer
Dim TabList As Variant

With Application
   '   Selecciona las columnas 1, 3 y 5
   .AddCustomList Array("Este", "Norte", "Oeste", "Sur", "Centro")
   '   Copia el contenido de la lista en una matriz
   NumList = .GetCustomListNum(Array("Este", "Norte", _
   "Oeste",  "Sur", "Centro"))
   TabList = .GetCustomListContents(NumList)
   '   Muestra el contenido de la lista en columna
   For i = LBound(TabList, 1) To UBound(TabList, 1)              .../...
```


Copyrighted Material

```
.../...
        Cells(i, 1) = TabList(i)
    Next i
End With
```

c. Selección de columnas no consecutivas

```
Dim MultipleRange As Range
'   Crea un objeto Range formado por las columnas 1,3 y 5
Set MultipleRange = Application.Union(Cells(1, 1), _
     Cells(1, 3), Cells(1, 5)).EntireColumn
'   Pone en negrita las celdas y las selecciona
MultipleRange.Font.Bold = True
MultipleRange.Select
```

d. Evaluación del resultado de una fórmula

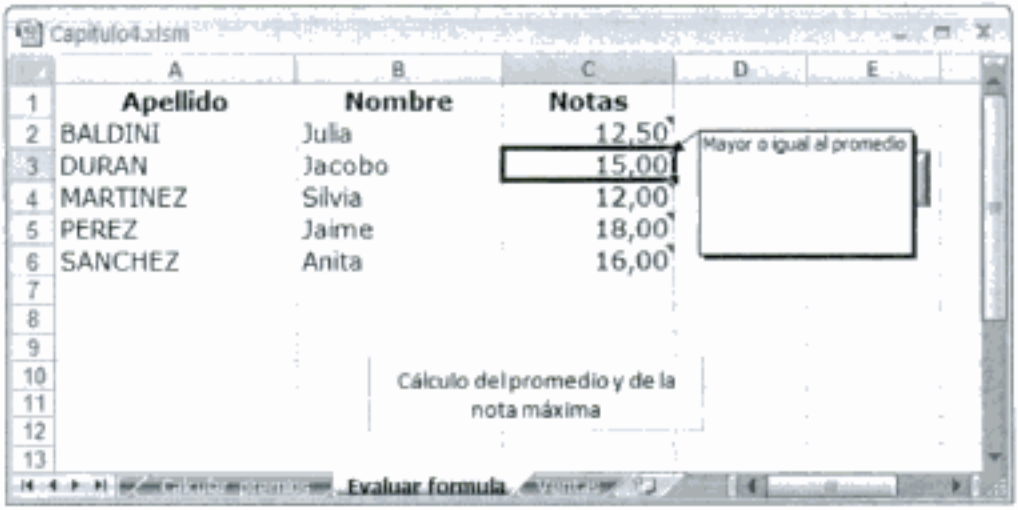
*Este ejemplo **calcula el promedio y el valor máximo de un rango de celdas** que contiene notas. Las notas se comparan, a continuación, con estos valores y se agrega un comentario para cada una.*

```
Sub Evaluacion()
   Dim dProm As Double
   Dim dMax As Double
   Dim Notas As Name
   Dim Celda As Range
   '   Selecciona del rango de celdas llamado Notas
   Set Notas = ThisWorkbook.Names("Notas")
   Notas.RefersToRange.Select
   '   Elimina los comentarios
   Selection.ClearComments
   '   Calcula el promedio y el máximo
   dProm = Evaluate("Average(notas)")
   dMax = Evaluate("Max(notas)")
   '   Muestra un comentario para cada nota
   For Each Celda In Selection
       With Celda
             Select Case .Value
                Case Is = dMax
                   .AddComment "El mejor"
                Case Is < dProm
                   .AddComment "Menor al promedio"          .../...
```

© Editions ENI - All rights reserved

Copyrighted Material

```
.../...
                    Case Else
                    .AddComment "Mayor o igual al promedio"
            End Select
        End With
    Next Celda

End Sub
```



B. Objeto Workbook

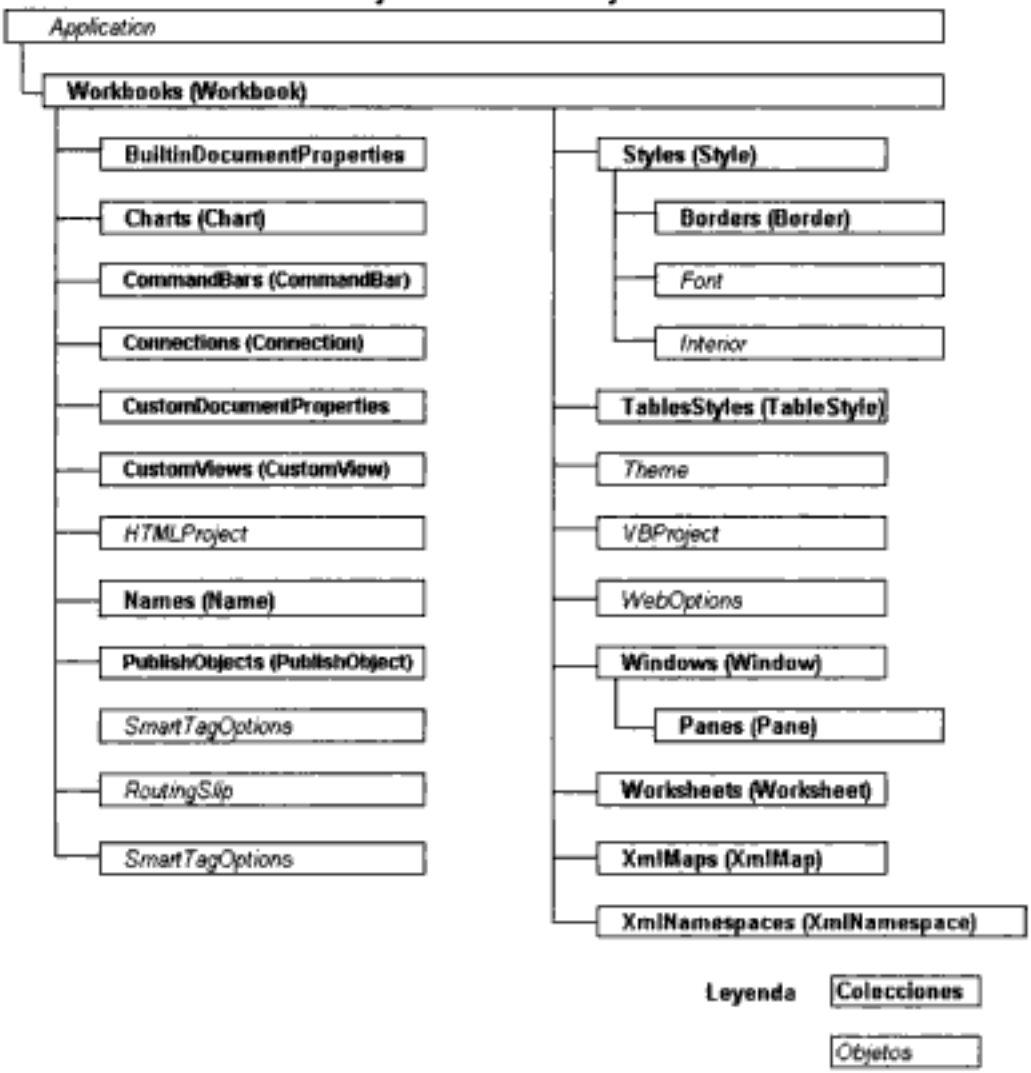
Este objeto representa un libro Microsoft Excel. El objeto Workbook es un miembro de la colección **Workbooks**.

El objeto es devuelto por las siguientes propiedades del objeto **Application**:

- Workbooks
- ActiveWorkbook
- ThisWorkbook

Copyrighted material

Extracto del modelo del objeto Excel - el objeto Workbook



1. Objetos y colecciones

Objetos

SmartTagOptions

Objeto que representa las opciones relativas a las etiquetas inteligentes.

Theme

Objeto que representa el tema aplicado al libro.

© Editions ENI - All rights reserved

Copyrighted material

VBProject

Objeto que representa el proyecto Visual Basic asociado a un libro.

WebOptions

Opciones relativas a la grabación y apertura de una página Web.

Colecciones

BuiltinDocumentProperties

Colección de las propiedades (autor, título, objeto, palabras clave, etc.) del libro.

Charts

Colección de los gráficos de un libro.

Connections

Colección de las conexiones a orígenes de datos para el libro.

CustomDocumentProperties

Colección de las propiedades de un libro (título, autor, comentarios, etc.).

CustomViews

Colección de las vistas personalizadas de un libro.

Names

Colección de los rangos con nombre de un libro.

PublishObjects

Colección de los elementos de un libro grabado como página Web y que pueden ser actualizados.

Styles

Colección de los estilos de un libro.

TableStyles

Colección de los distintos estilos aplicables a una tabla.

Copyrighted Material

Windows

Colección de las ventanas de la aplicación Excel.

Worksheets

Colección de las hojas de cálculo de un libro.

XmlMaps

Colección de los objetos **XmlMap** que se han agregado a un libro. Estos objetos se usan para administrar la relación entre los rangos de lista y los elementos de un esquema XML.

XmlNamespaces

Colección de los espacios de nombres XML contenidos en el libro especificado.

2. Propiedades

a. Propiedades relativas a la actualización y grabado de libros

CreateBackup

Boolean. Indica si se crea una copia de seguridad cuando se graba el archivo.

EnableAutoRecover

Boolean. Activa o desactiva la opción **Autorrecuperación**.

Saved

Boolean. Indica si el libro especificado no ha sido modificado después de la última grabación.

SaveLinkValues

Boolean. Indica si Microsoft Excel guarda los valores de los vínculos externos con el libro.

UpdateLinks

Constante. Parámetro del libro para la actualización de los vínculos OLE incorporados.

© Editions ENI - All rights reserved

Copyrighted material

UpdateRemoteReferences

Boolean. Indica si Microsoft Excel actualiza las referencias remotas del libro.

b. Propiedades relativas a libros compartidos

AutoUpDateFrequency

Entero largo. Devuelve o define el tiempo, en minutos, entre dos actualizaciones automáticas en el libro compartido. Si esta propiedad recibe el valor 0, la actualización solamente se hace al guardar el libro.

AutoUpDateSaveChanges

Boolean. Indica si las modificaciones realizadas al libro compartido se transmiten a los otros usuarios cuando el libro es actualizado automáticamente.

ChangeHistoryDuration

Entero largo. Devuelve o establece el número de días que se muestran en el historial de cambios del libro compartido.

ConflictResolution

Constante. Devuelve o define la forma en que se resuelven los conflictos cuando se actualiza un libro compartido.

HighlightChangesOnScreen

Boolean. Indica si las modificaciones en el libro compartido se resaltan en la pantalla.

KeepChangeHistory

Boolean. Indica si está habilitado el seguimiento de cambios en el libro compartido.

ListChangesOnNewsheet

Boolean. Indica si las modificaciones del libro compartido se muestran en una nueva hoja de cálculo.

MultiUserEditing

Boolean. Indica si el libro está abierto como una lista compartida.

Copyrighted material

RevisionNumber

Entero largo. Devuelve la cantidad de veces que el libro fue grabado mientras estaba abierto como lista compartida.

ShowConflictHistory

Boolean. Indica si la hoja de cálculo **Historial de conflictos** está visible en el libro abierto como lista compartida.

UserStatus

Variant. Devuelve una matriz de dos dimensiones indexada a partir de 1, con la información de cada usuario que tenga abierto el libro como lista compartida.

HasPassword

Boolean. Indica si el libro está protegido con contraseña.

Password

Cadena de caracteres. Devuelve o define la contraseña necesaria para abrir el libro.

PasswordEncryptionAlgorithm

Cadena de caracteres. Devuelve el algoritmo que usa Microsoft Excel para encriptar las contraseñas del libro.

PasswordEncryptionProvider

Cadena de caracteres. Devuelve el nombre del proveedor del algoritmo de cifrado que utiliza Microsoft Excel para encriptar las contraseñas del libro especificado.

PasswordEncryptionKeyLength

Entero largo. Indica la longitud de la clave del algoritmo que utiliza Microsoft Excel para encriptar las contraseñas del libro especificado.

PasswordEncryptionFileProperties

Boolean. Indica si Microsoft Excel encripta las propiedades de archivo del libro.

ProtectStructure

Boolean. Indica si el orden de las hojas de cálculo del libro está protegido.

© Editions ENI - All rights reserved

Copyrighted material

ProtectWindows

Boolean. Indica si las ventanas del libro están protegidas.

ReadOnly

Boolean. Indica si el libro fue abierto en modo de sólo lectura.

ReadOnlyRecommanded

Boolean. Indica si el libro fue grabado como recomendado sólo lectura.

RemovePersonalInformation

Boolean. Indica si la información personal del libro puede ser eliminada.

VBASigned

Boolean. Indica si el proyecto VBA del libro tiene firma digital.

WritePassword

Cadena de caracteres. Devuelve o define la contraseña de escritura para el libro.

WriteReserved

Boolean. Indica si el libro está protegido contra escritura.

WriteReservedBy

Cadena de caracteres. Indica el nombre del usuario que está autorizado a sobreescribir el libro.

c. Otras propiedades

Colors

Variant. Devuelve o define los colores de la paleta del libro. La paleta tiene 56 entradas, cada una representa un valor RGB.

Date1904

Boolean. Indica si el libro usa el sistema de fechas 1904.

Copyrighted Material

DefaultPivotTableStyle

Variant. Especifica el estilo de tabla de la colección **TableStyles** que se usa como estilo por defecto para las tablas dinámicas.

DefaultTableStyle

Variant. Especifica el estilo de tabla de la colección **TableStyles** que se usa por defecto.

DisplayDrawingObjects

Constante. Devuelve o define la manera en que se muestran las formas.

EnvelopeVisible

Boolean. Indica si son visibles el encabezado de composición de los mensajes de correo y la barra de herramientas de sobre.

FullNameURLEncoded

Cadena de caracteres. Nombre del libro, incluye su ruta en el disco.

HasVBProject

Boolean. Indica si un libro contiene código VBA. Esta propiedad es especialmente útil para determinar si un libro debe grabarse en un formato que acepte las macros.

IsAddin

Boolean. Indica si el libro se ejecuta como complemento.

PrecisionAsDisplayed

Boolean. Indica si los cálculos en el libro se realizan usando solamente los decimales visibles en las celdas.

ShowPivotChartActiveFields

Boolean. Indica o define si el panel de tareas filtro de gráfico dinámico es visible.

ShowPivotTableFieldList

Boolean. Indica si la lista de campos de una tabla dinámica puede mostrarse.

© Editions ENI - All rights reserved

Copyrighted material

TemplateRemoveExtData

Boolean. Indica si las referencias de datos externos se eliminan cuando el libro se graba como plantilla.

3. Lista de métodos

a. Métodos que actúan directamente sobre los libros

AddToFavorites

Agrega el libro especificado a la lista de favoritos de la barra de herramientas Web.

ApplyTheme

Aplica el tema especificado al libro.

Close

Cierra el libro especificado.

DeleteNumberFormat

Elimina del libro un formato númerico personalizado.

ExportAsFixedFormat

Publica un libro en formato PDF o XPS.

MergeWorkbook

Fusiona, en un libro abierto, los cambios realizados en otro libro.

NewWindows

Crea una copia de la ventana especificada.

OpenDatabase

Abre una base de datos y muestra la información en un nuevo libro. Devuelve un objeto **Workbook**.

Copyrighted Material

Post

Envía el libro especificado a una carpeta pública. Este método solamente funciona con un cliente Microsoft Exchange conectado a un servidor Microsoft Exchange.

PrintOut

Imprime el libro especificado.

PrintPreview

Muestra la vista preliminar del libro especificado.

PurgeChangeHistoryNow

Elimina las entradas del registro de cambios del libro especificado.

RefreshAll

Actualiza los rangos de datos externos y los informes de tablas dinámicas del libro especificado.

Route

Distribuye el libro usando la lista de distribución activa.

Save

Guarda las modificaciones del libro especificado.

SaveAs

Guarda el libro especificado en otro archivo (equivale a la opción **Guardar como** del menú **Archivo**).

SaveAsCopy

Guarda una copia del libro activo en un nuevo archivo sin modificar el libro abierto en la memoria.

UpdateFromFile

Actualiza un libro de sólo lectura a partir de la versión del libro grabado en el disco, si esta versión es más reciente que la copia del libro cargado en memoria. Si la copia del disco no fue modificada después que cargar el libro en memoria, la copia del libro residente en memoria no se recarga.

© Editions ENI - All rights reserved

Copyrighted Material

b. Métodos relativos a la seguridad

ChangeFileAccess

Modifica los permisos de acceso al libro que puede implicar la necesidad de cargar, desde el disco, una versión actualizada.

LockServerFile

Bloquea el libro en el servidor para evitar su modificación.

Protect

Protege el libro especificado para que no pueda modificarse.

ProtectSharing

Graba el libro e impide que sea compartido.

UnProtect

Quita la protección del libro especificado.

UnprotectSharing

Desactiva la protección que impide compartir el libro y lo graba.

c. Métodos relativos a libros compartidos

AcceptAllChanges

Acepta todas las modificaciones hechas al libro compartido especificado.

CanCheckIn

Devuelve una variable Boolean que indica si Excel puede extraer un libro especificado desde un servidor.

ExclusiveAccess

Atribuye al usuario actual un acceso exclusivo al libro abierto como lista compartida.

HighlightChangesOptions

Controla cómo se muestran los cambios en un libro compartido.

Copyrighted material

RejectAllChanges

Impide los cambios sobre el libro compartido especificado.

RemoveUser

Desconecta el usuario especificado del libro compartido.

d. Métodos relacionados con datos vinculados

BreakLink

Convierte las fórmulas vinculadas a otros orígenes Microsoft Excel u orígenes OLE en valores.

ChangeLink

Modifica un vínculo entre dos documentos.

EnableConnections

Activa las conexiones de datos en un libro.

FollowHyperlink

Muestra un documento de la caché, si ya ha sido transferido a la máquina local. De lo contrario, este método resuelve el hipervínculo, transfiere el documento destino a la máquina local y muestra el documento en la aplicación apropiada.

LinkInfo

Devuelve información acerca de la fecha y el estado de actualización del vínculo.

LinkSources

Devuelve una matriz de vínculos al libro. Los nombres de la matriz son los nombres de los documentos vinculados, ediciones o servidores DDE u OLE. Este método devuelve **Empty** si no hay vínculos en el libro.

OpenLinks

Abre los documentos origen de uno o más vínculos.

OpenXml

Abre un archivo XML en un nuevo libro. Devuelve un objeto **Workbook**.

© Editions ENI - All rights reserved

Copyrighted material

ReloadAs

Vuelve a cargar un libro basado en un documento HTML usando la codificación de documentos especificada.

SetLinkOnDate

Define el nombre de un procedimiento ejecutado a cada actualización de un vínculo DDE.

UpdateLink

Actualiza uno o más vínculos Microsoft Excel, DDE o OLE.

WebPagePreview

Muestra la vista previa del libro especificado, tal como se vería al ser guardado como página Web.

e. Métodos relativos al envío de libros

SendFaxOverInternet

Envía una hoja de cálculo como fax a los destinatarios especificados.

SendMail

Envía un mensaje de correo electrónico con el libro especificado.

SendForReview

Envía un mensaje de correo electrónico con el libro que hay que revisar a los destinatarios especificados.

EndReview

Termina la revisión de un archivo enviado para este fin con el método **SendForReview**.

ReplyWithChanges

Envía un mensaje de correo electrónico al autor de un libro enviado para revisión y le informa de que la revisión ha sido realizada.

Copyrighted Material

f. Otros métodos

Los métodos relativos a la importación y exportación de archivos al formato XML (SaveAsXMLData, XmlImport, etc.) se explican en el capítulo 11.

4. Ejemplos de códigos que usan el objeto Workbook

Para probar estos ejemplos, debe crear un directorio c:\Ventas con la base ejemplo de Access Northwind.mdb.

a. Creación de un libro Excel

El siguiente ejemplo permite:

- Cerrar todos los libros abiertos excepto el libro activo.
- Crear un nuevo libro.
- Proteger el libro con contraseña.
- Agregar el libro a la lista de Favoritos.
- Grabar y cerrar el libro.

```
Private Sub NuevoLibro()
Dim Libro As Workbook
Dim i As Integer
Dim j As Integer

'   Cierra los libros (excepto el libro activo)
'   y guarda los cambios

For Each Libro In Workbooks
    If Libro.Name <> ThisWorkbook.Name Then
       Libro.Close True
    End If
Next Libro

'   Crea un nuevo libro
Set Libro = Application.Workbooks.Add
With Libro
                                                    .../...
```

© Editions ENI - All rights reserved

Copyrighted material

```
.../...
'   Protege el libro con contraseña
   .Password = "Ventas"
   .WritePassword = "W_Ventas"
'   Guarda el libro
   .SaveAs "C:\Ventas\Ventas por región"
'   Agrega el libro a los favoritos de la barra de menú Web
   .AddToFavorites
'   Cierra el libro
   .Close
End With

End Sub
```

b. Importar una base de datos y exportarla al formato HTML

El siguiente ejemplo muestra cómo:

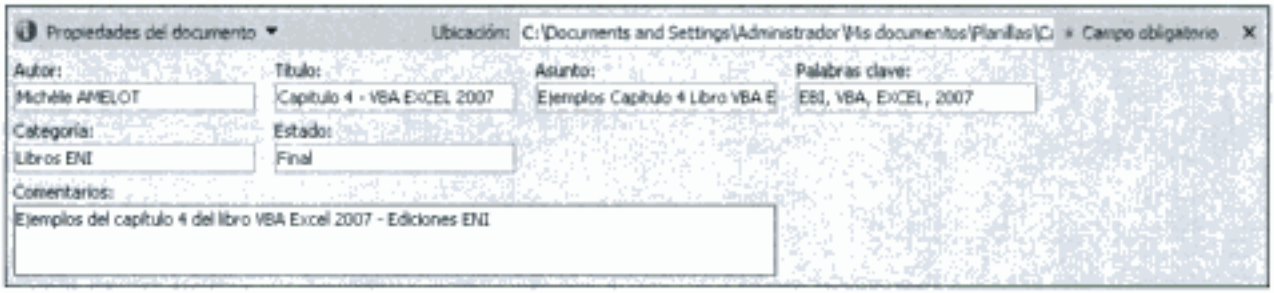
- Abrir la tabla Clientes del archivo "Northwind 2007.accdb" en un nuevo libro.
- Exportar esta información en un archivo HTML.
- Abrir el archivo HTML.

```
Sub CreaHTMLFile()
Dim Libro As Workbook

'   Importa la tabla Clientes de la base Access Northwind 2007
'   en un nuevo libro
Set Libro = Workbooks.OpenDatabase _
          (Filename:=ThisWorkbook.Path & "\Northwind 2007.accdb", _
          CommandText:="SELECT * FROM CLIENTS")
'   Exporta los clientes a un archivo Html
ActiveWorkbook.SaveAs Filename:=ThisWorkbook.Path & "\Clientes.htm", _
FileFormat :=xlHtml, ReadOnlyRecommended:=False, CreateBackup:=False
 '   Abre el archivo Html
Workbooks.Open Filename:=ThisWorkbook.Path & "\Clientes.htm"
End Sub
```


Copyrighted Material

c. Mostrar las propiedades de un libro

Este ejemplo muestra los nombres y valores de las diferentes propiedades del libro. Algunas de estas propiedades son accesibles al hacer un clic en el botón **Microsoft Office**, luego seleccionar **Preparar**, y después **Propiedades**.




```
Sub Propiedades()
   Dim i As Integer
   Dim m_ObjProp As Object
   '   Muestra la lista de propiedades del libro
   i = 1
   For Each m_ObjProp In ThisWorkbook.BuiltinDocumentProperties
      On Error Resume Next
      ActiveSheet.Cells(i, 2) = m_ObjProp.Value
      ActiveSheet.Cells(i, 1) = m_ObjProp.Name
      i = i + 1
   Next
End Sub
```

d. Exportar un libro al formato pdf

Para ejecutar este código, debe previamente ejecutar el programa de instalación del complemento Excel SaveAsPDFandXPS.exe (este programa se entrega con los ejemplos).

```
Sub Export_Pdf()
   '   Exporta el archivo al formato Pdf y lo abre en Acrobat Reader
   ThisWorkbook.ExportAsFixedFormat Type:=xlTypePDF, _
                                    Filename:="Capítulo4", _
                                    IncludeDocProperties:=True, _
                                    OpenAfterPublish:=True
End Sub
```

© Editions ENI - All rights reserved

Copyrighted material

C. El objeto Worksheet

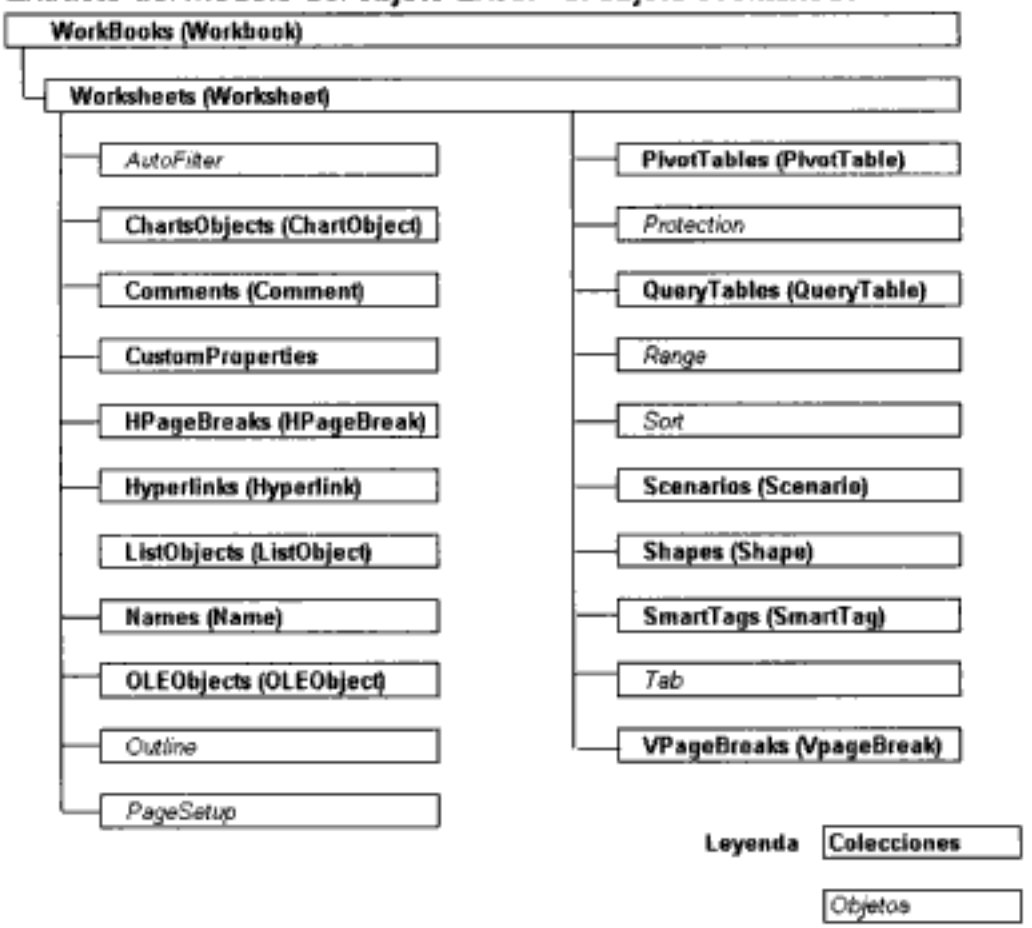
Este objeto representa una hoja de cálculo Excel. El objeto **Worksheet** es un miembro de la colección **WorkSheets** del objeto **Workbook**.

Este objeto es devuelto por las siguientes propiedades del objeto **Application**:

- Worksheets
- ActiveSheet

1. Lista de objetos y colecciones

Extracto del modelo del objeto Excel - el objeto Worksheet



Copyrighted material

2. Objetos y colecciones

Objetos

AutoFilter

Objeto que representa el autofiltro de la hoja de cálculo especificada.

Cells

Objeto **Range** que representa todas las celdas de la hoja de cálculo especificada.

OutLine

Objeto que representa el esquema de la hoja de cálculo especificada.

PageSetup

Objeto que representa las opciones de configuración de página de la hoja de cálculo especificada.

Protection

Objeto que representa las opciones de protección para la hoja de cálculo especificada. Estas opciones son accesibles en Excel a través del menú **Herramientas - Protección - Proteger hoja**.

Range

Objeto que representa una celda o un rango de celdas (una fila, una columna, etc.).

Sort

Objeto que representa el ordenamiento de un rango de datos.

Tab

Objeto que representa la ficha de la hoja de cálculo especificada.

Colecciones

ChartObjects

Colección de los gráficos incrustados en la hoja de cálculo especificada.

© Editions ENI - All rights reserved

Copyrighted Material

Comments

Colección de todos los comentarios de celda de la hoja de cálculo especificada.

CustomProperties

Colección de objetos **CustomProperty** que representa la información complementaria (metadatos para XML o etiquetas inteligentes).

HPageBreaks

Colección de los saltos de página horizontales en la zona de impresión de la hoja especificada.

Hyperlinks

Colección de los hipervínculos de la hoja de cálculo especificada.

ListObjects

Colección de las listas de la hoja de cálculo especificada.

Names

Colección de los rangos de celdas con nombre de la hoja de cálculo especificada.

OLEObjects

Colección de los objetos ActiveX y objetos OLE vinculados o incrustados en la hoja de cálculo especificada.

PivotTables

Colección de los informes de tabla dinámica de la hoja de cálculo especificada.

QueryTables

Colección de las tablas de hoja de cálculo creadas a partir de datos enviados desde un origen de datos externo.

Scenarios

Colección de los escenarios de la hoja de cálculo especificada.

SmartTags

Colección de las etiquetas inteligentes asociadas a las celdas de la hoja de cálculo.

Copyrighted material

VPageBreaks

Colección de los saltos de página verticales en la zona de impresión de la hoja especificada.

3. Propiedades

AutoFilterMode

Boolean. Indica si las flechas del menú desplegable de los autofiltros aparecen en la hoja de cálculo especificada.

ConsolidationFunction

Constante (**xlMax**, **xlMin**, **xlSum**, etc.). Devuelve la función usada para la consolidación actual.

ConsolidationOptions

Matriz de Boolean que representa las opciones relativas a la consolidación (rótulos en la fila superior, rótulos en la columna izquierda, vínculos con los datos de origen).

ConsolidationSources

Matriz de cadenas de caracteres que contiene los nombres de las hojas origen para la consolidación actual de la hoja de cálculo especificada.

DisplayPageBreaks

Boolean. Indica si se muestran los saltos de página (automáticos y manuales) de la hoja especificada.

DisplayRightToLeft

Boolean. Corresponde a la opción **Ver la hoja actual de derecha a izquierda** del cuadro de diálogo **Herramientas - Opciones** - ficha **Internacional**.

EnabledAutofilter

Boolean. Indica si las flechas del Autofiltro están activas solamente cuando está activa la protección de sólo interfaz de usuario.

EnableCalculation

Boolean. Indica si Excel recalcula automáticamente la hoja de cálculo cuando es necesario.

© Editions ENI - All rights reserved

Copyrighted Material

EnableFormatConditionsCalculation

Boolean. Devuelve o define si los formatos condicionales se aplican automáticamente si es necesario.

EnableOutLining

Boolean. Indica si los símbolos del esquema están activos cuando está activa la protección de sólo interfaz de usuario.

EnablePivotTable

Boolean. Indica si los controles y las acciones de la tabla dinámica están activos cuando está activa la protección de sólo interfaz de usuario.

EnableSelection

Constante. Devuelve o define los elementos que pueden seleccionarse en la hoja (xlNoRestrictions, xlNoSelection, xlUnlockedCells).

FilterMode

Boolean. Indica si hay aplicado un filtro a la hoja especificada.

MailEnvelope

Representa el encabezado de mensajes de correo electrónico para la hoja especificada.

Name

Cadena de caracteres que contiene el nombre de la hoja de cálculo.

ProtectContents

Boolean. Indica si el contenido de la hoja especificada está protegido.

ProtectDrawingObjects

Boolean. Indica si las formas gráficas están protegidas.

ProtectionMode

Boolean. Indica si está activa la protección de sólo interfaz de usuario.

ProtectScenarios

Boolean. Indica si están protegidos los escenarios de hoja de cálculo.

Copyrighted material

ScrollArea

Cadena de caracteres. Devuelve o establece el rango en que está permitido el desplazamiento de la hoja de cálculo especificada.

StandardHeight

Real doble. Devuelve el alto estándar (valor por defecto) de las filas de la hoja de cálculo especificada.

StandardWidth

Real doble. Devuelve el ancho estándar (valor por defecto) de las columnas de la hoja de cálculo especificada.

Type

Constante. Devuelve o define el tipo de la hoja de cálculo especificada (xlChart, xlDialogSheet, etc.).

Visible

Boolean. Indica si la hoja de cálculo especificada está visible.

4. Métodos

Activate

Activa la hoja de cálculo especificada. Equivale a hacer clic sobre la ficha de la hoja.

Calculate

Recalcula las celdas de la hoja de cálculo especificada.

CheckSpelling

Efectúa la verificación ortográfica de la hoja de cálculo especificada (equivale a la opción **Ortografía** del menú **Herramientas**).

CircleInvalid

Rodea con un círculo las entradas incorrectas en la hoja de cálculo especificada.

ClearArrows

Borra las flechas de auditoría de la hoja de cálculo especificada.

© Editions ENI - All rights reserved

Copyrighted material

ClearCircles

Borra los círculos que rodean las entradas incorrectas de la hoja de cálculo.

Copy

Hace una copia de la hoja de cálculo especificada (antes o después de una de las hojas del libro).

Delete

Elimina la hoja de cálculo especificada.

Evaluate

Calcula la expresión pasada como argumento y devuelve el resultado. La expresión debe corresponder a una fórmula de cálculo en inglés.

ExportAsFixedFormat

Publica una hoja de cálculo en formato PDF o XPS.

Move

Mueve la hoja de cálculo especificada a una posición dada (antes o después de una de las hojas del libro).

Paste

Pega el contenido del portapapeles en la hoja de cálculo especificada.

PasteSpecial

Pega el contenido del portapapeles en la hoja de cálculo especificada respetando el formato especificado (pegado especial).

PrintOut

Imprime la hoja de cálculo especificada.

PrintPreview

Muestra la vista preliminar de la hoja de cálculo especificada.

Protect

Protege la hoja de cálculo especificada.

Copyrighted Material

ResetAllPageBreaks

Redefine los saltos de página de la hoja de cálculo especificada.

SaveAs

Guarda la hoja de cálculo en un nuevo libro.

Select

Selecciona la hoja de cálculo.

SetBackgroundPicture

Define el gráfico de fondo de la hoja de cálculo especificada.

ShowAllData

Muestra todas las filas de la lista actualmente filtrada.

ShowDataForm

Muestra los datos de la hoja de cálculo especificada como formulario (corresponde a la opción **Formulario** del menú **Datos**).

UnProtect

Desactiva la protección de la hoja de cálculo especificada.

Los métodos relativos a mapas de datos XML (**XmlDataQuery, XmlMapQuery**, etc.) se explican en el capítulo 11.

5. Ejemplos de códigos que usan el objeto Worksheet

a. Ordenar las hojas de cálculo de un libro

El siguiente ejemplo permite:

- *Ordenar las hojas de un libro,*
- *Modificar el color de las fichas de cada hoja.*

© Editions ENI - All rights reserved

Copyrighted material

```
Sub OrganizarHojas()
   Dim Hoja As Worksheet
   Dim i As Integer

   '   Ordena las hojas del libro activo
   OrdenaHojas ActiveWorkbook

   '   Recorre las hojas de cálculo
   For i = 1 To ActiveWorkbook.Worksheets.Count
       Set Hoja = ActiveWorkbook.Worksheets(i)
       With Hoja
           '   Modifica el color de la ficha
           .Tab.Color = vbRed
       End With
   Next i

   End Sub

   Private Sub OrdenaHojas(Libro1 As Workbook)
   Dim i As Integer
   Dim j As Integer
   '   Procedimiento para ordenar alfabéticamente las hojas de cálculo
   '   de un libro
   With Libro1
       For i = 1 To .Worksheets.Count
           For j = 1 To i - 1
               If .Worksheets(i).Name < .Worksheets(j).Name Then
                   .Worksheets(i).Move before:=.Worksheets(j)
               End If
           Next j
       Next i
   End With

End Sub
```


Copyrighted Material

b. Protección de las hojas de cálculo de un libro

Este ejemplo protege las hojas de cálculo de un libro permitiendo las siguientes autorizaciones: formato de celdas, agregar columnas y filas, ordenamiento y auto-filtros. Las otras operaciones (eliminar filas o columnas, modificación de escenarios, etc.) quedan prohibidas.

```
Sub ProtegeHojas()
   Dim shtCurrent As Worksheet

   For Each shtCurrent In ActiveWorkbook.Worksheets
        shtCurrent.Protect Password:="Contraseña", _
          contents:=True, AllowFormattingCells:=True, _
          AllowInsertingColumns:=True, AllowInsertingRows:=True, _
          AllowSorting:=True, AllowFiltering:=True
   Next shtCurrent

End Sub
```

c. Ordenar una tabla

Este ejemplo ordena una tabla según tres columnas: ciudad, apellido y nombre.

```
Sub Ordena_Clientes()

    '   Ordena la tabla de clientes por ciudad, apellido y nombre
    Application.Goto Reference:="Clientes"
    With ActiveWorkbook.Worksheets("Clientes").Sort
         .SortFields.Clear
         .SortFields.Add Key:=Range("C2:C30"), SortOn:=xlSortOnValues, _
          Order:=xlAscending, DataOption:=xlSortNormal
         .SortFields.Add Key:=Range("B2:B30"), SortOn:=xlSortOnValues, _
          Order:=xlAscending, DataOption:=xlSortNormal
         .SortFields.Add Key:=Range("D2:D30"), SortOn:=xlSortOnValues, _
          Order:=xlAscending, DataOption:=xlSortNormal
          .SetRange Range("A1:G30")
          .Header = xlYes
          .MatchCase = False
          .Orientation = xlTopToBottom
          .Apply
    End With
End Sub
```

© Editions ENI - All rights reserved

Copyrighted material

D. El objeto Range

El objeto Range representa un rango de celdas y puede estar constituido por:

- una celda,
- una fila,
- una columna,
- un rango de celdas contiguas,
- un rango de celdas no contiguas,
- un rango 3D.

1. Propiedades y métodos que devuelven un objeto Range

Propiedades que devuelven un objeto Range

Propiedad	Objeto contenedor	Objeto devuelto
ActiveCell	Application Window	Objeto Range que representa la **primera celda** activa de la ventana activa o especificada.
Areas	Range	Colección que agrupa **todos los rangos** de una selección múltiple.
Cells	Application Range WorkSheet	Objeto Range que representa **una celda o una colección de celdas**: - de la hoja activa si el objeto contenedor es Application, - del rango especificado si el objeto contenedor es Range, - de la hoja de cálculo especificada si el objeto contenedor es Worksheet.

Copyrighted material

Propiedad	Objeto contenedor	Objeto devuelto
Columns	Application Range WorkSheet	Objeto Range que representa **las columnas**: - de la hoja activa si el objeto contenedor es Application, - del rango especificado si el objeto contenedor es Range, - de la hoja especificada si el objeto contenedor es Worksheet.
CurrentRegion	Range	Objeto Range que representa el objeto Range especificado, limitado por toda combinación de filas y columnas vacías.
EntireColumn		Objeto Range que representa **una o más columnas enteras** del rango especificado.
EntireRow		Objeto Range que representa **una o más filas enteras** del rango especificado.
End	Range	Objeto Range que representa **la celda situada al final de la zona** de rango especificado. Corresponde a la combinación de teclas [Fin][Flecha arriba], [Fin][Flecha abajo], [Fin][Flecha izquierda] o [Fin][Flecha derecha].
Offset	Range	Objeto Range especificado desplazado una o más filas o columnas.
Range	Application Worksheet Range	Objeto Range que representa **un rango de celdas**: - de la hoja activa si el objeto contenedor es Application, - del rango especificado si el objeto contenedor es Range, - de la hoja especificada si el objeto contenedor es Worksheet.

© Editions ENI - All rights reserved

Copyrighted Material

Propiedad	Objeto contenedor	Objeto devuelto
Rows	Application Range WorkSheet	Objeto Range que representa **todas las filas**: - de la hoja activa si el objeto contenedor es Application, - del rango especificado si el objeto contenedor es Range, - de la hoja especificada si el objeto contenedor es Worksheet.
UsedRange	Worksheet	Objeto Range que representa el rango usado en su totalidad por la hoja de cálculo especificada.

Métodos que devuelven un objeto Range

Método	Objeto contenedor	Objeto devuelto
Intersect	Application	Objeto Range que representa **la intersección** rectangular de varios rangos.
Union	Application	Objeto Range que representa **la unión** de varios rangos contiguos o discontinuos.

2. Sintaxis de las propiedades que devuelven un objeto Range

Cells

```
Objeto.Cells ([RowIndex],[ColumnIndex])
```

`RowIndex` número de fila de la celda.

`ColumnIndex` número de columna de la celda.

Si no se indica ningún argumento, Cells devuelve la colección de celdas del rango especificado.

Copyrighted material

Ejemplo

El siguiente ejemplo modifica el contenido y el color de celdas.

```
Sub LlenaHoja()
   Dim Celda As Range

   '   Modifica el contenido de la celda B1 de la hoja activa
   Application.Cells(1, 2) = "Enero"
   '   Modifica el contenido de la celda B2 de la hoja activa
   ActiveSheet.Range("A1:G10").Cells(2, 2) = "Febrero"
   '   Modifica el contenido de la celda B3 de la hoja activa
   ActiveSheet.Cells(3, 2) = "Marzo"
   '   Modifica el color de las celdas C1, C2, D1, D2
   For Each Celda In Range("C1:D2")
         Celda.Interior.Color = vbRed
   Next Celda
End Sub
```

Range

`Objeto.Range (Cell1,[Cell2])`

Donde `Cell1` y `Cell2` pueden ser:

- una celda (por ejemplo: "A1")
- un rango de celdas (por ejemplo: "A1:B7")
- un nombre de celda (por ejemplo: "Totales")

Si `Cell2` está especificado, `Range` devuelve un rango de celdas contiguas que incluye los dos rangos especificados.

© Editions ENI - All rights reserved

Copyrighted Material

Ejemplo

Este ejemplo crea la siguiente tabla en una hoja de cálculo.

Total1 | =SUMA(B4:B6)

	A	B	C	D	E	F
1		Resultados trimestrales				
2						
3		Este	Oeste	Sur	Norte	
4	enero					
5	febrero					
6	marzo					
7	Totales	0,00 €	0,00 €	0,00 €	0,00 €	

```
Sub HojaResultado()
   Dim i As Integer
   Dim Regiones

   With Application.ActiveSheet
      .Range("B1").Value = "Resultados trimestrales"

      '   Mes en columna
      For i = 1 To 3
          .Range("A" & i + 3).Value = _
          Format((DateValue("01/" & i & "/01")), "MMMM")
      Next i

      '   Regiones en fila
      Range("B3:E3").Value = Array("Este", "Oeste", "Sur", "Norte")
      '   Formato de celdas
      Range("B4:E7").NumberFormat = "# ##0.00 €"
      Range("A7").Value = "Totales"

      '   Nombra las celdas que contienen los totales
      '   Asigna una fórmula a las celdas con nombre
      Range("B7").Name = "Total1"
      Range("Total1").Formula = "=SUM(B4:B6)"
      Range("C7").Name = "Total2"
      Range("Total2").Formula = "=SUM(C4:C6)"
      Range("D7").Name = "Total3"
      Range("Total3").Formula = "=SUM(D4:D6)"
      Range("E7").Name = "Total4"
      Range("Total4").Formula = "=SUM(E4:E6)"
   End With

End Sub
```


Copyrighted material

OffSet

`Objeto.OffSet ([rowOffset],[columnOffset])`

`rowOffset`	Cantidad de filas de desplazamiento.
`colOffset`	Cantidad de columnas de desplazamiento.

`colOffset` y `rowOffset` pueden contener valores negativos.

Ejemplo

El siguiente ejemplo devuelve la dirección del rango que resulta de un desplazamiento de filas y columnas.

```
Sub DevuelveOffset()

   With Range("B5:C7")
      '   Desplazamiento de una fila hacia arriba
      '   Devuelve $A$5:$B$7
      MsgBox .Offset(0, -1).Address
      '   Desplazamiento de dos columnas hacia la derecha
      '   Devuelve $B$7:$C$9
      MsgBox .Offset(2, 0).Address
   End With

End Sub
```

Areas

`Objeto.Areas ([Index])`

`Index`	Número del rango en distintos rangos del objeto.

Si no se indica ningún argumento, `Areas` devuelve la colección de rangos especificada.

© Editions ENI - All rights reserved

Copyrighted material

Ejemplo

Este ejemplo permite:

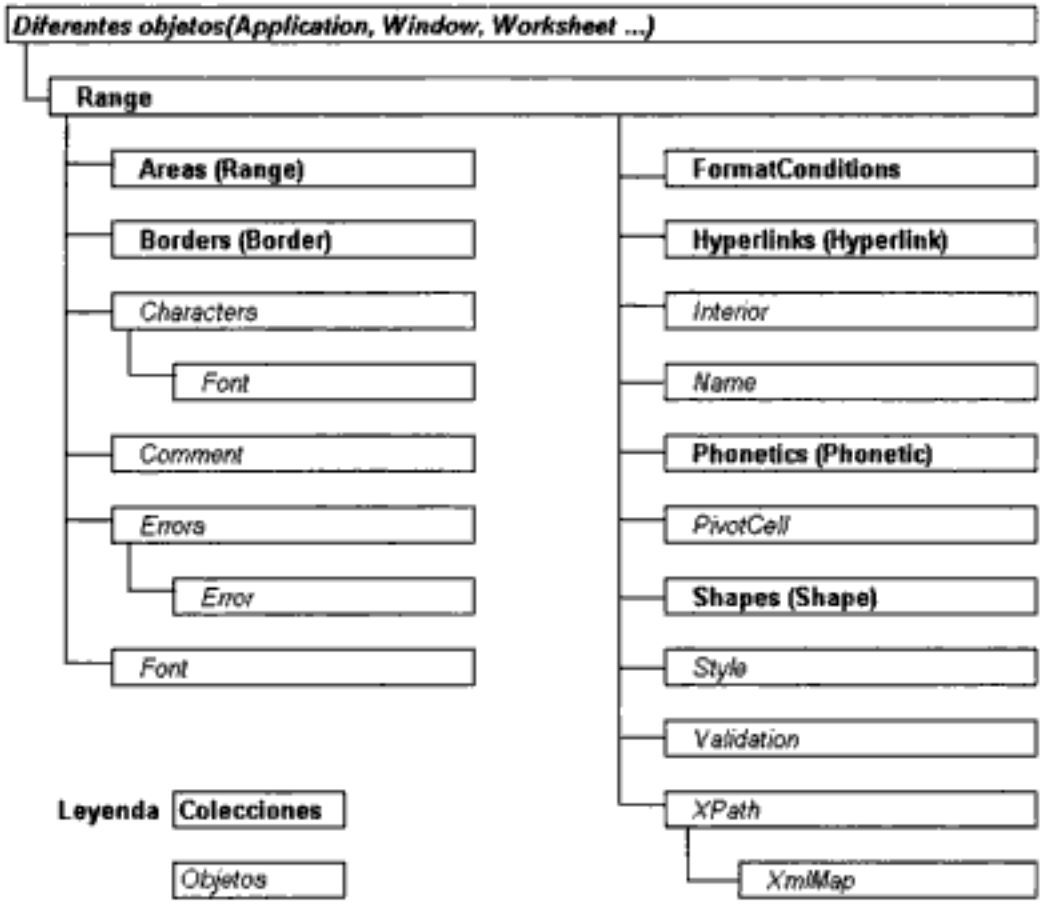
- *crear una zona formada por varios rangos de celdas discontinuos,*
- *llenar el primer rango a partir de una matriz,*
- *poner en negrita la fuente para todos los rangos.*

```
Sub MuchosRangos()
   Dim ZonaTot As Range
   Dim i As Integer

   '   Unión de muchos rangos discontinuos
   Set ZonaTot = Union(Range("B3:E3"), Range("B1:B5"),  _
         Range("J2:F6"))

   With ZonaTot
   '   Primer rango llenado a partir de una matriz
      .Areas(1).Value = Array("Este", "Oeste", "Sur", "Norte")
   '   Pone en negrita los caracteres de todos los rangos
      For i = 1 To 3
          .Areas(i).Font.Bold = True
      Next i
      .Select
   End With

End Sub
```


Copyrighted Material

3. Lista de objetos y colecciones

Extracto del modelo del objeto Excel - el objeto Range



Objetos

Characters

Objeto que representa los caracteres del texto de la celda especificada.

Comment

Objeto que representa el comentario asociado a la celda.

Errors

Objeto que representa los errores en el rango especificado.

© Editions ENI - All rights reserved

Copyrighted material

Font

Objeto que contiene los atributos de fuente (nombre, tamaño, color, etc.) del rango especificado.

Interior

Objeto que representa el relleno de las celdas del rango especificado.

PivotCell

Objeto que representa una celda en un informe de tabla dinámica.

Validation

Objeto que representa la validación de datos aplicada al rango especificado.

XPath

Objeto que representa un XPath (ruta XML) mapeado en el rango de celdas especificado.

Colecciones

Areas

Colección de todos los rangos en una selección de muchas zonas.

Borders

Colección de todos los bordes del rango de celdas especificado.

FormatConditions

Colección de los formatos condicionales del rango especificado.

HyperLinks

Colección de los hipervínculos del rango especificado.

Phonetics

Colección de objetos que contienen la información de una cadena de texto fonética específica en una celda.

Copyrighted Material

QueryTables

Colección de objetos que representa las tablas de hoja de cálculo creadas a partir de datos devueltos por un origen de datos externos.

SmartTags

Colección que representa las etiquetas inteligentes (Smart Tags) asignadas a cada celda del rango especificado.

4. Propiedades

a. Propiedades relacionadas con la posición y el formato de las celdas

AllowEdit

True si el rango puede ser modificado en una hoja de cálculo protegida.

AddressLocal

Devuelve la referencia del rango especificado en el idioma del usuario.

Address

Devuelve la referencia del rango en el lenguaje de la macro.

Column

Devuelve el número de la primera columna de la primera zona del rango especificado.

ColumnWidth

Devuelve o define el ancho de todas las columnas del rango especificado.

HorizontalAlignment

Constante. Define o devuelve el tipo de alineación horizontal.

IndentLevel

Devuelve o define el nivel de sangría efectivo.

Row

Devuelve el número de la primera fila de la primera zona del rango.

© Editions ENI - All rights reserved

Copyrighted material

RowHeight

Devuelve el alto, medido en puntos, de las filas del rango especificado.

UseStandardHeight

True si el alto de fila del objeto **Range** es igual al alto estándar de la hoja.

UseStandardWidth

True si el ancho de columna del objeto **Range** es igual al ancho estándar de la hoja.

VerticalAlignment

Constante. Define o devuelve el tipo de alineación vertical.

b. Propiedades que se relacionan con el contenido de celdas y con las fórmulas

CountLarge

Determina el valor máximo en el rango.

Formula

Devuelve o define la fórmula en el estilo de referencia A1.

FormulaLocal

Devuelve o define la fórmula del objeto usando las referencias de estilo A1 en el idioma del usuario.

FormulaR1C1

Devuelve o define la fórmula del objeto usando las notaciones de estilo R1C1.

FormulaR1C1Local

Devuelve o define la fórmula usando las notaciones de estilo R1C1 en el idioma del usuario.

PrefixCharacter

Devuelve el prefijo de alineación de la celda.

Copyrighted Material

Text

Valor de la celda especificada con el formato especificado (contenido visible de la celda).

Value

Valor de la celda especificada. Si la celda está vacía, la propiedad **Value** devuelve el valor **Empty** (use la función **IsEmpty** para testear este caso). Si el objeto **Range** contiene varias celdas, devuelve una matriz de valores (use la función **IsArray** para testear este caso).

WrapText

Boolean. **True** si Microsoft Excel inserta automáticamente retornos de carro en el texto del objeto.

c. Otras propiedades

MergeCells

True si el rango o el estilo contiene celdas combinadas.

ListHeaderRows

Devuelve la cantidad de filas de encabezado en el rango especificado.

5. Métodos

a. Métodos que devuelven un objeto

ColumnDifferences

Devuelve un objeto **Range** que representa las celdas donde el contenido es diferente al de la celda de comparación de cada columna.

Find

Busca una información específica en un rango y devuelve un objeto **Range** que representa la primera celda donde aparece esa información.

FindNext

Continúa una búsqueda (siguiente celda) iniciada con el método **Find**.

© Editions ENI - All rights reserved

Copyrighted material

FindPrevious

Continúa una búsqueda (celda anterior) iniciada con el método **Find**.

RowDifferences

Devuelve un objeto **Range** que representa las celdas en las que el contenido es diferente del de la celda de comparación de cada fila.

SpecialCells

Devuelve un objeto **Range** que representa las celdas que corresponden al tipo y al valor especificados.

SetPhonetic

Crea un objeto **Phonetic** para las celdas del objeto **Range** especificado.

b. Métodos que se relacionan con la presentación de las celdas

AddComment

Agrega un comentario al rango.

AutoFit

Modifica el ancho de las columnas del rango o el alto de las filas para ajustar sus datos.

BorderAround

Agrega un borde a un rango y define las propiedades **Color**, **LineStyle** y **Weight** del nuevo borde.

ClearComments

Quita todos los comentarios de celda del rango especificado.

ClearFormats

Elimina los formatos de las celdas.

ClearNotes

Borra las notas escritas y sonoras de todas las celdas del rango especificado.

Copyrighted material

InsertIndent

Inserta una sangría en las celdas del rango especificado.

Justify

Reorganiza el texto en un rango de forma que lo llene de manera uniforme.

Merge

Combina las celdas.

NoteText

Devuelve o define las notas de celdas asociadas a la celda ubicada en la esquina superior izquierda del rango.

Sort

Ordena un rango de valores.

TextToColumns

Redistribuye en varias columnas una columna de celdas que contiene texto.

UnMerge

Separa un celda combinada en celdas individuales.

c. Métodos que se relacionan con el contenido de las celdas

AutoFill

Ejecuta un llenado incremental en las celdas del rango especificado.

AutoComplete

Devuelve una coincidencia de la funcionalidad Autocompletar de la lista.

ClearContents

Borra el contenido de las celdas.

Consolidate

Consolida datos que provienen de varios rangos situados en diferentes hojas de cálculo en un único rango situado en una única hoja de cálculo.

© Editions ENI - All rights reserved

Copyrighted material

Copy

Copia al portapapeles el objeto **Range** del rango especificado.

CopyFromRecordSet

Copia el contenido de un objeto **Recordset** ADO o DAO en una hoja de cálculo, comenzando en la esquina superior izquierda del rango especificado.

CopyPicture

Copia el objeto seleccionado en el portapapeles en forma de imagen.

Cut

Corta el objeto y lo guarda en el portapapeles o lo pega en un destino especificado.

Delete

Elimina las celdas e indica cómo reemplazar las celdas eliminadas.

FillDown

Rellena un rango hacia abajo.

FillLeft

Rellena un rango hacia la izquierda.

FillRight

Rellena un rango hacia la derecha.

FillUp

Rellena un rango hacia arriba.

FunctionWizard

Inicia el Asistente para funciones en la celda situada en la esquina superior izquierda del rango.

Insert

Inserta celdas e indica cómo desplazar las celdas.

Copyrighted material

Parse

Redistribuye un rango de datos y lo divide en varias celdas. Distribuye el contenido del rango de manera que llene varias columnas adyacentes; el rango no puede tener más de una columna de ancho.

PasteSpecial

Efectúa el pegado especial de un objeto **Range** proveniente del portapapeles en el rango especificado.

RemoveDuplicates

Elimina los datos repetidos en un rango de valores.

Replace

Busca y reemplaza caracteres en las celdas del rango especificado. El uso de este método no cambia la selección ni la celda activa.

d. Métodos que se relacionan con los nombres de celdas

ApplyNames

Define nombres para las celdas del rango especificado.

CreateNames

Crea nombres en el rango especificado en función de los rótulos de texto de la hoja.

ListNames

Pega una lista con los nombres de la hoja de cálculo que no están ocultos, comenzando por la primera celda del rango.

e. Métodos que se relacionan con los filtros

AdvancedFilter

Filtra o copia los datos de una lista en función de una zona de criterios.

AutoFilter

Filtra una lista.

© Editions ENI - All rights reserved

Copyrighted Material

f. Métodos que se relacionan con el modo esquema

ApplyOutlineStyles

Aplica los estilos del esquema al rango especificado.

AutoOutline

Crea automáticamente un esquema para el rango especificado. Si el rango tiene una sola celda, Microsoft Excel crea un esquema para toda la hoja.

Group

En un esquema, aumenta el nivel del rango en el esquema. El rango debe ser una fila o una columna entera o un rango de filas o de columnas.
Para un rango discontinuo de un informe de tabla dinámica, reagrupa el rango.
Para una única celda del rango de datos de un campo de tabla dinámica, realiza un reagrupamiento numérico o cronológico en el campo.

Ungroup

Promueve un rango en un esquema (es decir, reduce su nivel de esquema). El rango especificado debe ser una fila o una columna o bien un rango de filas o columnas. Si el rango se encuentra en un informe de tabla dinámica, el método desagrupará los elementos incluidos en el rango.

g. Métodos que se relacionan con la herramienta de Auditoría

NavigateArrow

Desplaza una flecha de rastreo del rango especificado hacia la o las celdas precedentes, dependientes o que provocan un error.

ShowDependents

Muestra las flechas de rastreo que señalan las celdas dependientes directas del rango.

ShowPrecedents

Muestra las flechas de rastreo que señalan las celdas precedentes directas del rango.

Copyrighted Material

ShowErrors

Muestra las flechas de rastreo a través de la estructura en árbol de las celdas precedentes a la celda que origina el error y devuelve el rango que la contiene.

h. Otros métodos

Calculate

Calcula las fórmulas de todos los libros abiertos.

CalculateRowMajorOrder

Calcula el rango de celdas indicado a partir de su esquina superior izquierda y hasta la esquina inferior derecha en el orden fila-campo.

Dirty

Indica que el rango especificado se recalculará en la próxima actualización de la hoja.

ExportAsFixedFormat

Publica los datos de un rango de valores en formato PDF o XPS.

PrintOut

Imprime el rango de celdas.

Run

Ejecuta una macro.

Table

Crea una tabla de datos a partir de los valores de entrada y de las fórmulas definidas en una hoja de cálculo.

© Editions ENI - All rights reserved

Copyrighted Material

E. Ejemplos de uso de los objetos

1. Cálculo del importe de un premio

CALCULO DE PREMIOS ANUALES

Apellido	Nombre	VN	Premio
MARTIN	Silvia	€ 124.560	€ 500
DURÁN	Jacobo	€ 145.890	€ 1.000
SANCHEZ	Anita	€ 151.239	€ 2.000
MARTINEZ	Jose	€ 389.000	€ 3.000
BALDINI	Julia	€ 133.950	€ 1.000
RONCAOLI	Sergio	€ 89.000	€ -
MULLER	Petter	€ 900.000	€ 3.000
PFEIFER	Martha	€ 88.650	€ -
DURANT	Anne	€ 152.010	€ 2.000
	Total	€ 2.174.299	€ 12.500
	Promedio	€ 241.589	€ 1.389

Calcular premios

Calcular_premios

El rango de celdas "D6:D14" debe recibir el nombre VN.

Cuando el usuario hace clic en el botón de comando **Calcular premios** se ejecutará el procedimiento **Calc_Premios**. Este procedimiento selecciona el rango de celdas llamado VN (celdas "D6:D14") y llama a la función **Premio** para calcular la prima y asignarla a la celda de la derecha.

```
Sub Calc_Premios()
   Dim dblVNProm As Double
   Dim celda As Range

   '    Selección del rango llamado VN
   ThisWorkbook.Names("VN").RefersToRange.Select
   '    Cálculo del promedio de la selección
   dblVNProm = Evaluate("AVERAGE(VN)")                    .../...
```


Copyrighted material

```
.../...
   '    Recorre las celdas de la selección
   '    La prima calculada se asigna a la celda de la derecha
   For Each celda In Selection
        Cells(celda.Row, celda.Column + 1) = _
        Premio(celda.Value, dblVNProm)
   Next celda
End Sub
```

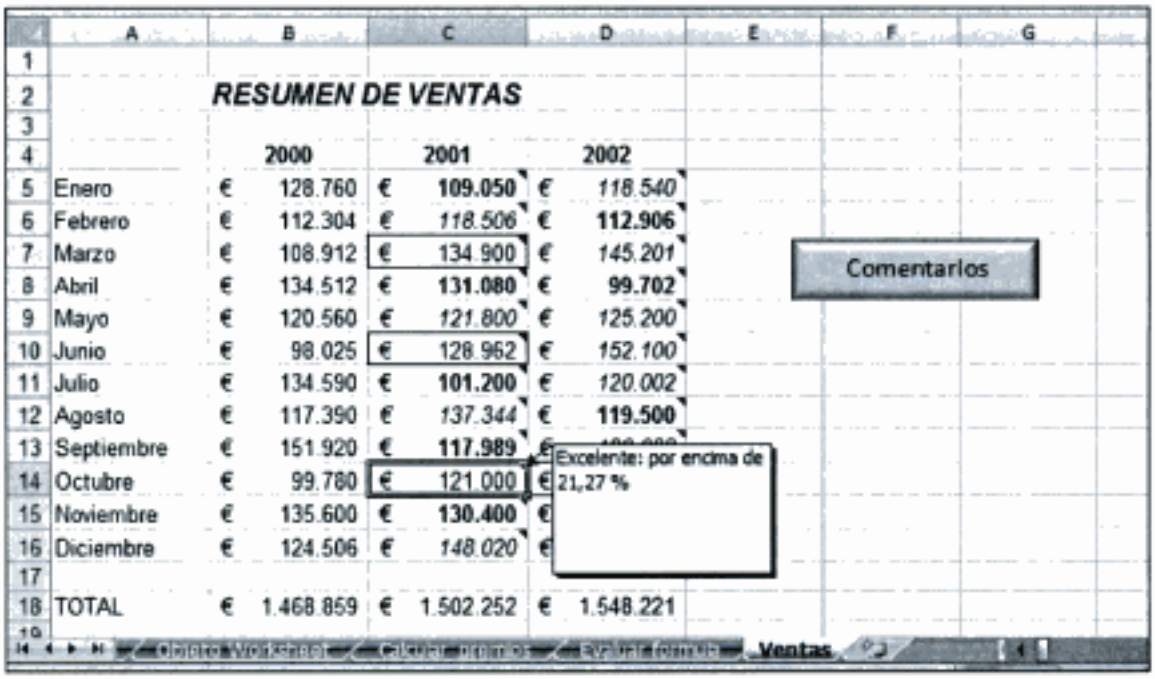
La función **Premio** calcula el premio en función del VN (valor negociado) y del promedio de los otros VN.

```
Function Premio(dblVN As Double, dblVNProm As Double) As Double

   '    Premio en función del importe VN
   Select Case dblVN
      Case Is < 100000
           Premio = 0
      Case Is < 125000
           Premio = 500
      Case Is < 150000
           Premio = 1000
      Case Else
           Premio = 2000
   End Select

   '    Si el VN es superior al promedio
   '    premio extra de 1000
   If dblVN > dblVNProm Then
      Premio = Premio + 1000
   End If

End Function
```

© Editions ENI - All rights reserved

Copyrighted material

2. Asignación de comentarios a las celdas



Cuando el usuario hace clic en el botón **Comentarios**, se ejecuta el procedimiento **Mostrar_Comentarios**. Este procedimiento llama al procedimiento **Compara_Valor** para comparar cada una de las celdas seleccionadas con la celda situada a su izquierda.

```
Sub Mostrar_Comentarios()
   Dim rng1 As Range
   Dim rng2 As Range
   Dim rngCurrent As Range
   Dim col As Object
   Dim i As Integer
   Dim j As Integer
   '    Borra los comentarios y los estilos de la selección actual
   Set rngCurrent = ThisWorkbook.Worksheets("Ventas").Range("C5:D16")
   With rngCurrent
       .ClearComments
       .Font.Bold = False
       .Font.Italic = False
       .Borders.LineStyle = xlLineStyleNone
   '    Recorre las columnas seleccionadas
   '    Compara el valor de cada celda de la columna
   '    con el de la celda situada a su izquierda
       For i = 1 To .Columns.Count                                  .../...
```


Copyrighted Material

```
.../...
            Set col = .Columns(i)
            For j = 1 To col.Cells.Count
               Set rng1 = col.Cells(j)
               Set rng2 = Cells(rng1.Row, rng1.Column - 1)
               Compara_Valor rng1, rng2
           Next j
       Next i
   End With

End Sub
```

Al llamar a la función **Compara_Valor** recibe como argumentos las celdas que hay que comparar. En función del porcentaje de evolución (negativo, < 20%, > 20%), se asigna un comentario y un formato a la primera celda.

```
Sub Compara_Valor(rng1 As Range, rng2 As Range)
   Dim dbl1, dbl2, dbl3 As Double
   Dim strEvol As String

   '   Compara los valores de dos celdas y asigna un comentario
   With rng1
      dbl1 = rng2.Value
      dbl2 = .Value
   dbl3 = (dbl2 - dbl1) / dbl1
      strEvol = Format(Abs(dbl3), "0.00 %")
      Select Case dbl3
         Case Is < 0
             .Font.Bold = True
             .AddComment "Atención: por debajo de " & strEvol
         Case Is < 0.2
             .Font.Italic = True
             .AddComment "Bien: por encima de " & strEvol
         Case Else
             .Borders.LineStyle = xlContinuous
             .AddComment "Excelente: por encima de " & strEvol
       End Select
   End With

End Sub
```

© Editions ENI - All rights reserved

Copyrighted Material

Copyrighted Material

Capítulo 5: Cuadros de diálogo

A. Presentación 180

B. Cuadros de diálogo integrados 180

1. El objeto Dialog 180

2. Los métodos GetOpenFileName y GetSaveAsFileName . . . 181

C. Cuadros de diálogo predefinidos 184

1. La función InputBox 184

2. El método InputBox 185

3. La función MsgBox 186

4. Constantes usadas en los cuadros de diálogo 190

Copyrighted material

A. Presentación

El objetivo principal de los cuadros de diálogo es controlar el intercambio de información con el usuario: mostrar mensajes, pedir información, vista y entrada de datos, etc.

Existen tres tipos de cuadros de diálogo:

- los cuadros de diálogo llamados **cuadros de diálogo integrados**, que permiten, por ejemplo, abrir o guardar un archivo, definir las opciones de Excel, imprimir hojas de cálculo, ordenar datos, etc.
- los cuadros de diálogo **predefinidos**, que permiten mostrar un mensaje, hacer una pregunta al usuario o invitarlo a introducir una información,
- los cuadros de diálogo **personalizados** o **formularios**, que permiten mostrar o introducir datos en una interfaz amigable. La creación de formularios personalizados se explica en el capítulo siguiente.

B. Cuadros de diálogo integrados

1. El objeto Dialog

Los cuadros de diálogo integrados son objetos **Dialog** pertenecientes a la colección **Dialogs** del objeto **Application**.

Sintaxis

→ Para mostrar un cuadro de diálogo, use el método **Show** según la siguiente sintaxis: `Application.Dialogs(xlDialog).Show`

donde `xlDialog` es una constante Excel que indica el cuadro de diálogo que hay que mostrar.

Copyrighted Material

Ejemplos de constantes xlDialog

Constante	Cuadro de diálogo
xlDialogBorder	Bordes
xlDialogFontProperties	Fuente
xlDialogDisplay	Opciones de visualización
xlDialogDefineName	Definir un nombre
xlDialogFormulaGoto	Ir a...
xlDialogOpen	Abrir
xlDialogSaveAs	Guardar como...
xlDialogSort	Ordenar

2. Los métodos GetOpenFileName y GetSaveAsFileName

Los métodos **GetOpenFileName** y **GetSaveAsFileName** del objeto **Application** muestran, respectivamente, los cuadros de diálogo **Abrir...** y **Guardar como...** del menú **Archivo**.

A diferencia de los objetos **Dialogs** correspondientes (constantes xlOpen y xlSaveAs), estos métodos no realizan ninguna acción; solamente permiten recuperar el nombre del archivo introducido o seleccionado por el usuario.

Sintaxis del método GetOpenFileName

```
Application.GetOpenFileName(FileFilter, FilterIndex, Title,
ButtonText, MultiSelect)
```

Todos los argumentos son opcionales.

`FileFilter`	Criterios de filtrado: nombre del filtro seguido de la extensión. Por ejemplo: "PáginaWeb (*.htm;*.html) ,*.htm;*.html".
`FileIndex`	Índice del criterio de filtrado por defecto.
`Title`	Título del cuadro de diálogo.
`ButtonText`	Etiqueta del botón **Abrir** (solamente para Macintosh).
`MultiSelect`	Indica si el usuario puede seleccionar varios archivos.

© Editions ENI - All rights reserved

Copyrighted material

Sintaxis del método GetSaveAsFileName

```
Application.GetSaveAsFileName(InitialeFile, FileFilter,
FilterIndex, Title, ButtonText)
```

Todos los argumentos son opcionales.

`InitialeFile` nombre del archivo que aparece en la zona de texto **Nombre**. Si se omite este argumento, Excel usa el nombre del libro activo.

Los otros argumentos son los mismos que los del método **GetOpenFileName**.

Ejemplo

Este ejemplo permite:

- mostrar el cuadro de diálogo **Abrir** con la posibilidad de seleccionar varios archivos,
- guardar en una matriz los nombres de los archivos seleccionados con extensión xls y que no se encuentren ya abiertos,
- mostrar un mensaje que indica los archivos que hay que abrir,
- abrir esos archivos tras pedir confirmación.

```
Sub AbreLibros()
   Dim strFiles
   Dim xlFiles
   Dim blnAbierto As Boolean
   Dim strMensaje As String
   Dim wbk As Workbook
   Dim i As Integer
Dim j As Integer

   '   Muestra el cuadro de diálogo Abrir
   strFiles = Application.GetOpenFilename _
            (filefilter:="Archivos Excel (*.xlsm),*.xlsm", _
            Title:="Seleccione los archivos que hay que abrir", _
            MultiSelect:=True)                                .../...
```


Copyrighted Material

```
.../...
   '   Prueba si los archivos han sido seleccionados
   If TypeName(strFiles) = "Variant()" Then
      ReDim xlFiles(UBound(strFiles))
      For i = 1 To UBound(strFiles)
          '   Controla la extensión del archivo
          If Right(strFiles(i), 3) = "xls" Then
             '   Prueba si el archivo ya está abierto
             blnAbierto = False
             For Each wbk In Workbooks
                If wbk.Path & "\" & wbk.Name = strFiles(i) Then
                    blnAbierto = True
                 End If
             Next wbk
             '   Guarda el nombre del archivo en una matriz
             If Not blnAbierto Then
                j = j + 1
                xlFiles(j) = strFiles(i)
                strMensaje = strMensaje & strFiles(i) & vbCr
             End If
          End If
      Next i

      '   Abre todos los archivos Excel tras confirmación
      If j > 1 Then
       strMensaje = "Confirme la apertura de los archivos: " _
                    & vbCr & strMensaje
         If MsgBox(strMensaje, vbYesNo + vbQuestion) = vbYes Then
            For i = 1 To j
                Workbooks.Open Filename:=xlFiles(i)
            Next i
         End If
      End If
   Else
      MsgBox "Ningún archivo seleccionado"
   End If
End Sub
```

© Editions ENI - All rights reserved

Copyrighted Material

C. Cuadros de diálogo predefinidos

1. La función InputBox

Muestra una pregunta y devuelve el texto escrito por el usuario.

InputBox(prompt,title,default,xpos,ypos,helpfile,context)

prompt	Cadena que aparecerá como mensaje.
title	Cadena que aparecerá en la barra de título.
default	Valor tomado por defecto.
xpos	Posición horizontal del cuadro de diálogo (expresado en twips).
ypos	Posición vertical del cuadro de diálogo (expresado en twips).
helpfile	Nombre del archivo de ayuda contextual.
context	Número del contexto de ayuda.

Ejemplo

Este ejemplo muestra un cuadro de diálogo que pide el nombre de las celdas que hay que borrar (las celdas reciben el nombre de cada mes).

```
Sub Borrar_Celdas_Nombradas()
   Dim opcion As String
   '   Pide introducir el mes que hay que borrar
   '   Si el mes se reconoce, borra las celdas con nombre
   '   Si no, muestra un mensaje de error
   opcion = InputBox( _
           Prompt:="¿Qué mes borra?", _
           Title:="Borrar celdas")
   On Error GoTo 1
   Application.Goto reference:=opcion
   Selection.Clear
   Exit Sub
   1 MsgBox "No puede borrar, nombre de celda inexistente"
End Sub
```


Copyrighted Material

2. El método InputBox

Actúa como la función `InputBox` pero permite controlar el tipo de datos que hay que introducir.

```
objeto.InputBox(prompt,title,default,left,top,helpfile,
helpContextID,type)
```

El objeto es obligatorio, y debe ser un objeto **Application**.

`prompt`	Mensaje mostrado.
`title`	Título del cuadro de diálogo.
`default`	Valor tomado por defecto.
`left`	Posición horizontal del cuadro de diálogo (en puntos).
`top`	Posición vertical del cuadro de diálogo (en puntos).
`helpfile`	Nombre del archivo de ayuda en línea.
`help ContextID`	Número del contexto de ayuda.
`type`	Tipo de datos que se devolverá:
0	Fórmula
1	Número
2	Cadena
4	Valor Boolean
8	Referencia de celda
16	Valor de error
64	Matriz de valores en una selección de celdas.

© Editions ENI - All rights reserved

Copyrighted material

Para aceptar varios tipos de datos, haga la suma de los valores. Por ejemplo, si la zona de entrada puede aceptar un texto o un número, indique el valor 3 (1 + 2) como tipo.

Ejemplo

Pide al usuario seleccionar la o las celda(s) que hay que pintar.

```
Sub Celdas_A_Pintar()
   Dim strRep as Range
   '   Si el usuario selecciona celdas,
   '   éstas se pintan de rojo
   '   Si hace clic en Cancelar, termina el procedimiento
   On Error GoTo 1
   Set strRep = Application.InputBox( _
        Prompt:="Seleccione la o las celda(s) que hay que pintar", _
        Title:="Celda que hay que pintar", Default:="A1",  Type:=8)
       strRep.Interior.ColorIndex = 3
1 End Sub
```

3. La función MsgBox

Esta función muestra un mensaje en un cuadro de diálogo. Puede incluir un icono, y de uno a tres botones.

Sintaxis de la instruction

Usada cuando aparece un único botón.

```
MsgBox <message> [, [<type>][, <title>]]
```

Sintaxis de la función

Usada cuando aparece más de un botón. Permite saber qué botón activó el usuario, a través del valor devuelto.

```
MsgBox (<message> , [<buttons>][, <title>]
[,helpfile, context])
```

`message` Texto del mensaje que aparece en el cuadro de diálogo.

Copyrighted Material

`buttons` Expresión numérica que representa la suma de los valores que especifican los botones que hay que mostrar, el tipo de icono que hay que usar, la identidad del botón por defecto, y la modalidad del cuadro.

`title` Texto en la barra de título.

`helpfile` Archivo de ayuda que hay que usar.

`Context` Número del contexto de ayuda.

Valores del argumento Buttons

Constante simbólica	Valor	Significado
Número y tipo de botón		
vbOKOnly	0	Muestra solamente el botón **Aceptar**.
vbOKCancel	1	Muestra los botones **Aceptar** y **Cancelar**.
vbAbortRetryIgnore	2	Muestra los botones **Anular**, **Reintentar** e **Ignorar**.
vbYesNoCancel	3	Muestra los botones **Sí**, **No** y **Cancelar**.
vbYesNo	4	Muestra los botones **Sí** y **No**.
vbRetryCancel	5	Muestra los botones **Reintentar** y **Cancelar**.
vbMsgBoxHelpButton	16384	Muestra un botón de ayuda.
Tipo de icono		
vbCritical	16	Muestra el icono .
vbQuestion	32	Muestra el icono .
vbExclamation	48	Muestra el icono .
vbInformation	64	Muestra el icono .
Botón por defecto		
vbDefaultButton1	0	Primer botón.

© Editions ENI - All rights reserved

Copyrighted Material

Constante simbólica	Valor	Significado
vbDefaultButton2	256	Segundo botón.
vbDefaultButton3	512	Tercer botón.
vbDefaultButton4	768	Cuarto botón.
Modalidad		
vbApplicationModal	0	Aplicación modal. El usuario debe responder al mensaje que aparece en el cuadro de mensajes antes de seguir trabajando en la aplicación actual.
vbSystemModal	4096	Sistema modal. Se suspenden todas las aplicaciones hasta que el usuario responda al mensaje que aparece en el cuadro de mensajes.
Presentación		
vbMsgBoxSetForeground	65536	Muestra la ventana del cuadro de mensaje en primer plano.
vbMsgBoxRight	524288	Alinea el texto a la derecha.
vbMsgBoxRtlReading	1048576	Especifica el orden de lectura de derecha a izquierda para los sistemas hebreo y árabe.

Los valores devueltos también se definen mediante constantes:

Constante	Valor devuelto	Botón elegido
vbOK	1	**Aceptar**
vbCancel	2	**Cancelar**
vbAbort	3	**Anular**
vbRetry	4	**Reintentar**
vbIgnore	5	**Ignorar**

Copyrighted Material

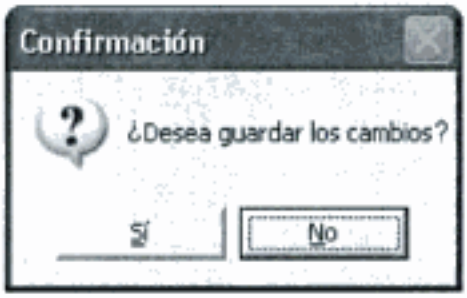
Constante	Valor devuelto	Botón elegido
vbYes	6	**Sí**
vbNO	7	**No**

Ejemplos: uso de la función MsgBox

```
StrRep = MsgBox ("Por favor, confirme", 292, _
"Confirmación")
```

o

```
StrRep = MsgBox ("¿Desea guardar los cambios?", _
vbYesNo + vbQuestion + vbDefaultButton2, _
"Confirmación")
```



Uso de la instrucción MsgBox

```
MsgBox "Resultado incorrecto",vbCritical,"Error"
```

© Editions ENI - All rights reserved

Copyrighted material

Ejemplo

Pide al usuario que seleccione las columnas que se deben eliminar (la selección de columnas puede hacerse a partir de una o más celdas de la columna), y que confirme la eliminación.

```
Sub Confirmacion()
   Dim CualCol As Range
   Dim AConfirmar
   '   Elegir las columnas
   Set CualCol = Application.InputBox( _
       Prompt:="Seleccione las columnas que desea eliminar", _
       Title:="Elija las columnas", _
       Type:=8)
   Set CualCol = CualCol.EntireColumn
   '   Selecciona las columnas y pide confirmación
   CualCol.Select
   AConfirmar = MsgBox( _
          Prompt:="Confirme la eliminación de las columnas", _
          Title:="Supresión de las columnas", _
          Buttons:=vbYesNo + vbExclamation + vbDefaultButton2)
   If AConfirmar = vbYes Then CualCol.Delete
End Sub
```

4. Constantes usadas en los cuadros de diálogo

En los mensajes de los cuadros de diálogo, puede usar las siguientes constantes para insertar algunos caracteres especiales.

Caracteres que desea insertar	Constantes	Equivalentes
Retorno de carro y salto de línea	vbCrLf	Chr(13) + Chr(10)
Salto de párrafo	vbCr	Chr(13)
Salto de línea	vbLf	Chr(10)
Caracter nulo	vbNullChar	Chr(0)
Diferente a una cadena de longitud nula	vbNullString	Cadena que tiene el valor cero
Tabulación	vbTab	Chr(9)
Retroceso	vbBack	Chr(8)

Copyright Material

Estas constantes pueden usarse en otras instrucciones además de los cuadros de diálogo.

Ejemplo

Para mostrar este cuadro de diálogo:

se utilizó el siguiente procedimiento:

```
Sub Identificacion()
   MsgBox Prompt:="Usted es:" & vbCr & vbTab _
               & Application.UserName _
               & vbCrLf & vbTab & Application.OrganizationName
End Sub
```

© Editions ENI - All rights reserved

Copyrighted material

Copyrighted Material

Capítulo 6: Formularios

A. Presentación 194

B. Crear un formulario 194

C. Personalizar un formulario 204

1. Escribir procedimientos 204
2. Lista de eventos 204
3. Ejecución y cierre de un formulario 209

D. Ejemplo de formulario personalizado 211

1. Presentación 211
2. Código asociado al botón macro de la ficha Empleados . . 213
3. Código VBA asociado al formulario 213

Copyrighted material

A. Presentación

Los formularios (también llamados cuadros de diálogo personalizados, formularios personalizados, hojas del usuario o UserForm) permiten disponer de **interfaces de usuario simples y amigables** para la introducción, la modificación o la visualización de datos.

Los formularios personalizados son cuadros de diálogo sobre los que se puede:

- **Ubicar controles ActiveX**, tales como cuadros de entrada de texto, listas desplegables, botones de comando, etc.
- **Asociar código VBA** para responder a distintos eventos del usuario (clic sobre un botón de comando, introducción en una zona de texto, selección en una lista desplegable, etc.).

B. Crear un formulario

Un formulario se crea en una hoja **UserForm**.

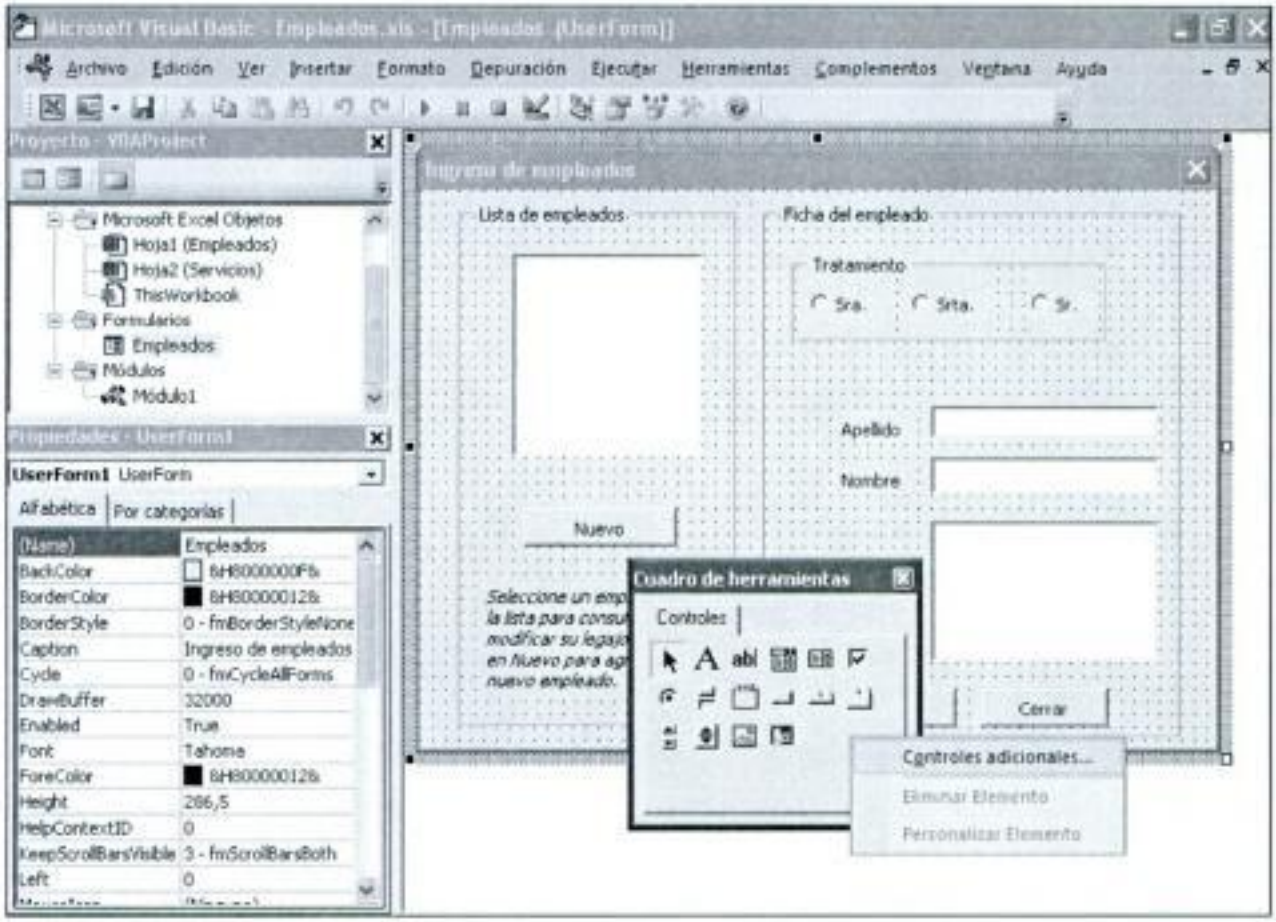
→ Para insertar una hoja UserForm, acceda a Microsoft Visual Basic, y luego seleccione las opciones **Insertar - UserForm**.

Se agregará una hoja llamada **UserForm n** (por ejemplo: UserForm1), aparecerá un formulario vacío y el cuadro de herramientas.

→ Para mostrar la ventana de propiedades, seleccione las opciones:

Ver
Ventana propiedades [F4]

Copyrighted material



(Name)

Nombre del formulario.

Caption

Texto en la barra de título.

→ Para dimensionar el formulario, selecciónelo y arrastre los controladores de tamaño o indique las propiedades **Height** y **Width** del formulario.

La opción **Controles adicionales** permite agregar otros controles desde la barra de herramientas.

© Editions ENI - All rights reserved

Copyrighted material

Lista de controles

Herramientas	Nombres	Objetos
	Etiqueta	Label
	Cuadro de texto	TextBox
	Cuadro combinado	ComboBox
	Cuadro de lista	ListBox
	Casilla de verificación	CheckBox
	Botón de opción	OptionButton
	Botón de alternar	ToggleButton
	Marco	Frame
	Botón de comando	CommandButton
	Barra de tabulaciones	TabStrip
	Página múltiple (selección de páginas)	Multipage
	Barra de desplazamiento	ScrollBar
	Botón de número (selección de valores)	SpinButton
	Imagen	Image
	RefEdit (selección de rangos)	RefEdit

Crear un control

→ Seleccione el control que desea crear y arrastre el puntero para definir un área rectangular.

Copyrighted material

→ Al soltar el botón del ratón, aparecerá el control y la herramienta **Seleccionar objetos** ( ) pasará a ser la herramienta activa.

Para dibujar varios controles del mismo tipo, haga doble clic en la herramienta correspondiente.

Descubrir algunas propiedades

(Name)

Especifica el nombre del control.

Caption

Indica el texto de una etiqueta.

ControlTipText

Crea una etiqueta informativa.

Visible

Especifica si un control está oculto o visible.

Enabled

Determina si el foco puede estar sobre el control.

Value

Define el estado o el contenido de un control.

ControlSource

Vincula un control a una celda (cuadro de texto) o un rango de celdas (cuadro de lista).

© Editions ENI - All rights reserved

Copyrighted material

Determinar el acceso a un control

- Para definir el orden de tabulación, seleccione las opciones:
  Ver
  Orden de tabulación

- Para desactivar la posibilidad de usar la tecla [Tab] para acceder a un control, seleccione el control e indique **False** en la propiedad **TabStop**.

- Para asignar una tecla de acceso rápido, seleccione el control e indique la tecla de acceso en la propiedad **Accelerator**.

Si el acceso rápido se aplica a un control **Label**, el control que sigue al **Label** en el orden de tabulación recibirá el foco y no el control **Label** propiamente dicho.

Dimensionar un control

- Para modificar el tamaño de un control, seleccione los controles y arrastre el controlador de tamaño o seleccione el control e indique las propiedades **Height** y **Width** que determinan el alto y el ancho del control en puntos.

- Para uniformizar los tamaños, seleccione los controles que desea dimensionar.
  Formato
  Igualar tamaño
  En función del tamaño deseado, elija **Ancho**, **Alto** o **Ambos**.

- Para ajustar el tamaño, seleccione los controles que desea ajustar y seleccione las opciones **Formato - Ajustar tamaño al contenido** o **Ajustar tamaño a la cuadrícula**.

Ubicar un control

- Para definir la posición de un control, seleccione el control que desea mover y arrastre el ratón o seleccione el control y asigne las propiedades **Left** y **Top** que indican la distancia entre el control y el borde izquierdo y superior del formulario.

Copyrighted Material

→ Para alinear controles entre ellos, seleccione los controles que desea alinear.

Formato
Alinear

Según el control de referencia, elija: **Izquierda**, **Centro**, **Derecha**, **Superior**, **Medio**, **Inferior** o **A la cuadrícula**.

→ Para administrar el espaciado entre controles, seleccione los controles.

Formato
Espacio horizontal o **Espacio vertical**

En función del espacio deseado, seleccione las opciones **Igualar**, **Aumentar**, **Disminuir** o **Quitar**.

→ Para centrar un control en el formulario, seleccione el control.

Formato
Centrar en el formulario Horizontalmente o **Verticalmente**

Aplicar formato

Asigne las siguientes propiedades:

Font

Define la tipografía.

BackColor

Especifica el color de fondo.

ForeColor

Especifica el color de primer plano.

BorderColor

Especifica el color del borde.

BorderStyle

Especifica el tipo de borde.

© Editions ENI - All rights reserved

Copyrighted material

Hidden page

Hidden page

Resumen de propiedades por objeto

	CheckBox	ComboBox	Command Button	Frame	Image	Label	ListBox	MultiPage	OptionButton	ScrollBar	SpinButton	TabStrip	TextBox	Toggle Button	RefEdit
(Name)	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
Accelerator	X		X			X			X					X	
AutoSize	X	X	X		X	X			X				X	X	X
AutoTab		X											X		X
AutoWordSelect		X											X		X
BackColor	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
BorderColor		X		X	X	X	X						X		X
BorderStyle		X		X	X	X	X						X		X
Caption	X		X	X		X			X				X		
ControlSource	X	X					X		X	X	X		X	X	
ControlTipText	X	X	X		X	X	X	X	X	X	X	X	X	X	X
DragBehavior		X											X		X
Enabled	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
EnterKeyBehavior													X		X
Font	X	X	X	X		X	X	X	X			X	X	X	X
ForeColor	X	X	X	X		X	X	X	X	X	X	X	X	X	X
Height	X	X	X		X	X	X	X	X	X	X	X	X	X	X
HideSelection		X											X		X
IntegralHeight							X						X		X
Left	X	X	X		X	X	X	X	X	X	X	X	X	X	X
Locked	X	X	X				X		X				X	X	X

Copyrighted material

Hidden page

C. Personalizar un formulario

1. Escribir procedimientos

→ Para mostrar la ventana de código de un control:
haga doble clic en el control para el que desee asignar un código, o seleccione el control, y seleccione las opciones:

Ver [F7]
Código

→ Para insertar un nuevo evento, abra la lista de la derecha y seleccione el evento deseado.

Si no indica un evento, el evento sugerido para la mayoría de los controles es el evento **Click**.

→ Para volver a mostrar un control, seleccione las opciones:
Ver [Mayús] [F7]
Objeto

2. Lista de eventos

Activate

Ocurre cuando se activa la hoja.

AddControl

Ocurre cuando se inserta un control en una hoja.

AfterUpDate

Ocurre tras modificar datos.

BeforeDragOver

Ocurre cuando se está ejecutando una operación de arrastrar y soltar.

Copyrighted Material

BeforeDropOrPaste

Ocurre cuando el usuario está a punto de colocar o pegar datos en un objeto.

BeforeUpDate

Ocurre antes de la modificación de datos.

Change

Ocurre cuando se modifica la propiedad **Value**.

Click

Ocurre cuando el usuario hace clic en un control o cuando selecciona definitivamente un valor para un control con más de un valor posible.

DblClick

Ocurre cuando el usuario hace doble clic.

DeActivate

Ocurre cuando la hoja deja de ser la ventana activa.

DropButtonClick

Ocurre cada vez que se muestra o se oculta una lista desplegable.

Enter

Ocurre antes de que un control reciba realmente el foco desde un control de la misma hoja.

Error

Ocurre cuando un control detecta un error y no puede devolver información del error al programa que lo ha llamado.

Exit

Ocurre inmediatamente antes de que un control pierda el foco en favor de otro control de la misma hoja.

Initialize

Ocurre después de que se carga un objeto, pero antes de que se muestre.

© Editions ENI - All rights reserved

Copyrighted material

Hidden page

SpinUp

Ocurre cuando el usuario hace clic en la flecha superior o derecha del contador.

Terminate

Ocurre tras la descarga de la hoja.

Zoom

Ocurre cuando cambia el valor de la propiedad **Zoom**.

Resumen de eventos por objeto

	CheckBox	ComboBox	Command Button	Frame	Image	Label	ListBox	Multipage	OptionButton	ScrollBar	SpinButton	TabStrip	TextBox	ToggleButton	UserForm	RefEdit
Activate															X	
AddControl				X				X							X	
AfterUpDate	X	X					X		X	X	X		X	X	X	X
BeforeDragOver	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
BeforeDropOrPaste	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
BeforeUpDate	X	X					X		X	X	X		X	X		X
Change	X	X					X	X	X	X	X	X	X	X		X
Click	X	X	X	X	X	X	X	X	X			X		X	X	
DblClick	X	X	X	X	X	X	X	X	X			X	X	X	X	X
DeActivate															X	
DropButtonClick		X											X			X
Enter	X	X	X	X			X	X	X	X	X	X	X	X		X
Error	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

© Editions ENI - All rights reserved

Copyrighted material

Hidden page

Ejemplo

Si la fecha introducida es incorrecta, se cancela el evento Exit: el cursor queda situado en la zona de texto.

```
Private Sub txtFechaFin_Exit(ByVal Cancel As _
MSForms.ReturnBoolean)

If IsNull(txtFechaFin) Then Exit Sub
'  La fecha debe ser correcta
If Not IsDate(txtFechaFin) Then
   MsgBox "Fecha incorrecta", vbCritical
   Cancel = True
   Exit Sub
End If

'  La fecha de fin debe ser = fecha de inicio
If DateValue(txtFechaFin) < DateValue(txtFechaIni) Then
   MsgBox "La fecha de fin es anterior a la fecha de inicio", vbCritical
   Cancel = True
   Exit Sub
End If
End Sub
```

Solamente los eventos BeforeDragOver, BeforeDropOrPaste, BeforeUpdate, DblClick, Exit, Error y QueryClose tienen un argumento Cancel. Los demás eventos no pueden cancelarse.

3. Ejecución y cierre de un formulario

→ Para ejecutar un formulario desde de la hoja UserForm, seleccione las opciones:

Ejecutar
Ejecutar Sub/UserForm [F5]

→ Para ejecutar un formulario desde un módulo, use el método **Show** o la instrucción **Load**.

© Editions ENI - All rights reserved

Copyrighted material

➔ Show (método)

Sintaxis

`ObjetoUserForm.Show`

Muestra el objeto **UserForm** indicado.

➔ Load (instrucción)

Sintaxis

`Load ObjetoUserForm`

Carga el objeto sin mostrarlo.

➔ Para cerrar un formulario, use el método **Hide** o la instrucción **Unload**.

➔ Hide (método)

Sintaxis

`ObjetoUserForm.Hide`

Oculta el formulario sin descargarlo.

➔ Unload (instrucción)

Sintaxis

`Unload ObjetoUserForm`

Elimina el formulario de la memoria.

Copyrighted Material

Eventos invocados

Los métodos e instrucciones de ejecución y de cierre de formularios desencadenan los siguientes eventos:

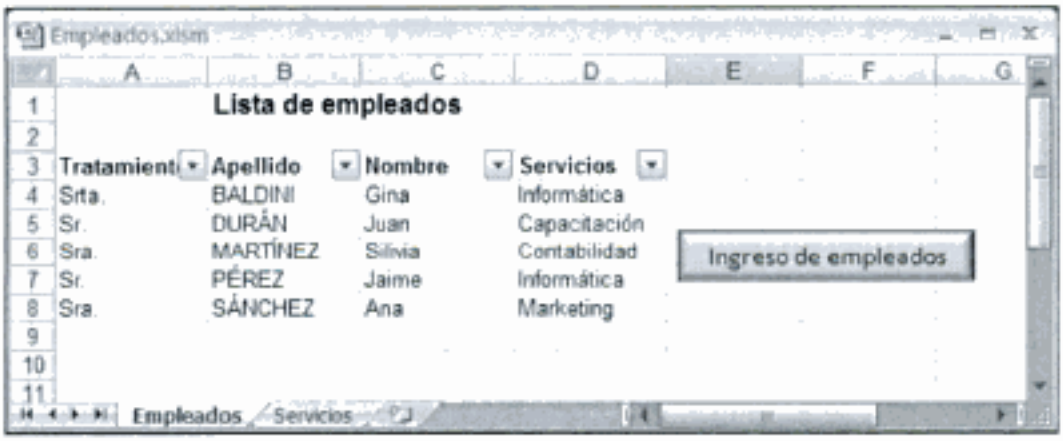
Método o instrucción	Eventos
Show	Initialize
	Activate
Load	Initialize
Hide	Sin evento
Unload	QueryClose
	Terminate

D. Ejemplo de formulario personalizado

1. Presentación

Este ejemplo muestra cómo crear un formulario personalizado para introducir o modificar las fichas "empleados".
El libro Empleados.xls contiene dos hojas de cálculo y un formulario.

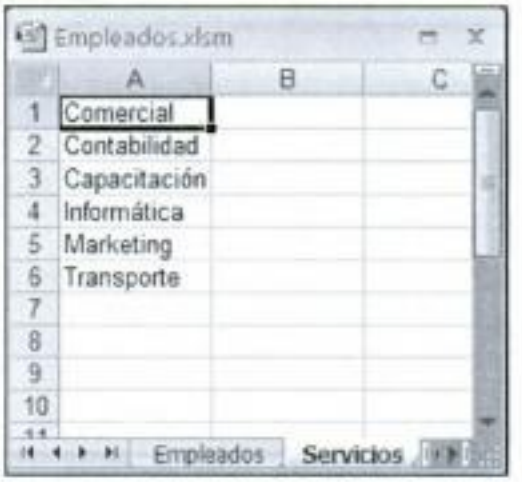
La hoja "Empleados" contiene la lista de empleados:



© Editions ENI - All rights reserved

Copyrighted material

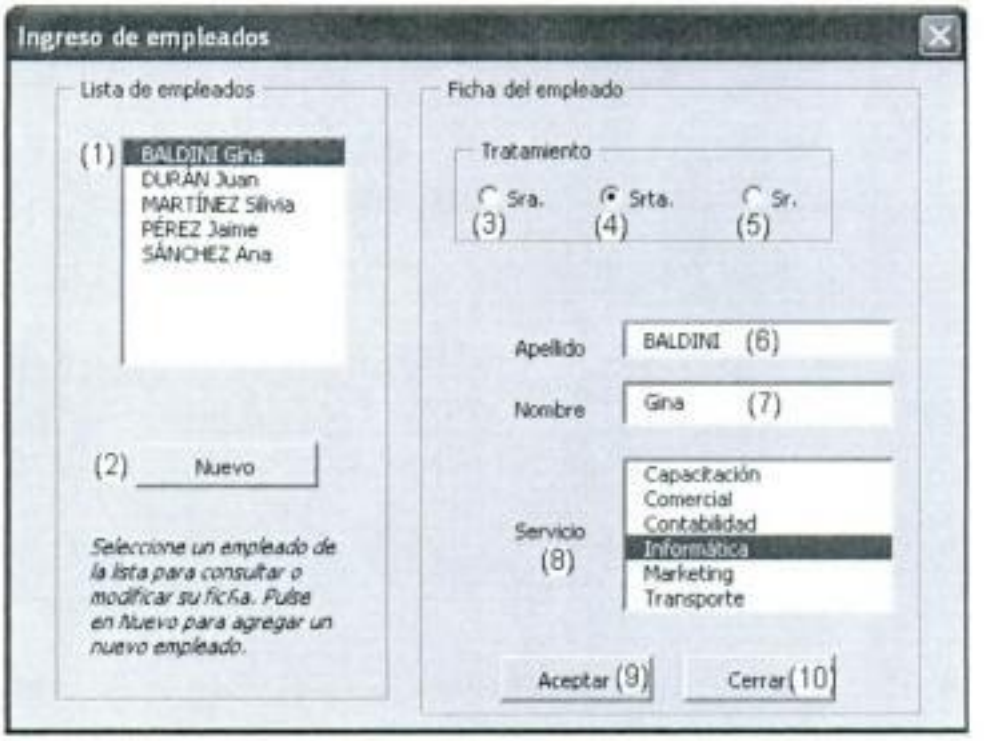
La hoja "Servicios" contiene la lista de servicios:



El formulario Empleados permite:

- modificar la información de un empleado y actualizar la hoja Excel Empleados,
- crear un nuevo empleado y agregarlo a la lista de la hoja Excel Empleados.

La lista de servicios se lee desde la hoja Servicios.



Copyrighted material

Lista de controles del formulario Empleados:

N°	Tipo de control	Nombre
1	Cuadro de lista	lstEmpleados
2	Botón de comando	cmdNuevo
3	Botón de opción	optSra
4	Botón de opción	optSrta
5	Botón de opción	optSr
6	Cuadro de texto	txtApellido
7	Cuadro de texto	txtNombre
8	Cuadro de lista	lstServicios
9	Botón de comando	cmdAceptar
10	Botón de comando	cmdCerrar

2. Código asociado al botón macro de la ficha Empleados

Este código está contenido en el módulo de clase ThisWorkbook.

```
Sub Mostrar_Formulario()
'   Muestra el formulario Empleados
Empleados.Show
End Sub
```

3. Código VBA asociado al formulario

```
Option Explicit
' Nombre de la aplicación
Const strAppName = "Introducción de empleados"
Dim bNuevo As Boolean
_____________________________________________________________
Private Sub UserForm_Initialize()
   Dim rng As Range
   Dim cell As Range
   ' Muestra la lista de servicios
   With ThisWorkbook.Worksheets("Servicios")
        .Activate
         Set rng = .Range("A1").CurrentRegion
        ' Ordena los servicios por orden alfabético
         rng.Sort Key1:   =Range("A1")
         lstServicios.Clear
         For Each cell In rng                                .../...
```

© Editions ENI - All rights reserved

Copyrighted Material

```
.../...
            If cell.Text <> "" Then
               lstServicios.AddItem cell.Text
            Else
               Exit For
            End If
         Next cell
   End With
   ' Muestra la lista de empleados
   Mostrar_Empleados
   lstEmpleados.ListIndex = 0
   ' Nuevo empleado por defecto
   bNuevo = True
End Sub
```

```
Private Sub Mostrar_Empleados()
   Dim rng As Range
   Dim linea As Range
   ' Muestra la lista de empleados
   With ThisWorkbook.Worksheets("Empleados")
        .Activate
        Set rng = .Range("A3").CurrentRegion
        Set rng = .Range("A4:   D" & rng.Rows.Count + 3)
        lstEmpleados.Clear
        For Each linea In rng.Rows
            If Cells(linea.row, 2) <> "" Then
               lstEmpleados.AddItem Cells(linea.row, 2) & " " & _
               Cells(linea.row, 3)
            Else
               Exit For
            End If
        Next linea
   End With
End Sub
```

```
Private Sub cmdAceptar_Click()
   Dim rng As Range
   Dim i As Integer
    ' Control de datos introducidos
   If txtNombre = "" Or txtApellido = "" _
      Or IsNull(lstServicios) Then
      MsgBox "Nombre, apellido y servicio obligatorio", _
             vbExclamation, strAppName
      txtNombre.SetFocus
      Exit Sub
   End If                                                    .../...
```


Copyrighted Material

```
.../...
   If Not (optSr Or optSra Or optSrta) Then
      MsgBox "Tratamiento obligatorio", vbExclamation, strAppName
      optSr.SetFocus
      Exit Sub
   End If
   With ThisWorkbook.Worksheets("Empleados")
       '  Agrega el empleado en la primera fila vacía
       If bNuevo Then
          Set rng = .Range("A3").CurrentRegion
          i = rng.Rows.Count + 3
       Else
       '  Modificación del empleado seleccionado
          i = lstEmpleados.ListIndex + 4
       End If
       If optSr Then
          .Cells(i, 1) = "Sr."
       ElseIf optSra Then
           .Cells(i, 1) = "Sra."
       Else
           .Cells(i, 1) = "Srta."
       End If
       .Cells(i, 3) = Empleados.txtNombre
       .Cells(i, 2) = Empleados.txtApellido
       .Cells(i, 4) = Empleados.lstServicios
       '   Ordenar empleados
       Ordenar_Empleados
   End With
   '   Muestra la lista de empleados
      If bNuevo Then Mostrar_Empleados
      Inicializa_Empleados
End Sub
```

```
Private Sub cmdCerrar_Click()
   '   Pide confirmación y cierra el formulario
   If MsgBox("¿Desea terminar el ingreso?", _
      vbQuestion + vbYesNo, strAppName) = vbYes Then
   Unload Me
   End If
End Sub
```

```
Private Sub cmdNuevo_Click()
   '   Inicializa la ficha Empleados
   Inicializa_Empleados
   bNuevo = True
End Sub
```

.../...

© Editions ENI - All rights reserved

Copyrighted material

```
.../...
Private Sub Ordenar_Empleados()
   Dim rng As Range
   '   Ordena la lista de empleados por apellido y nombre
   Set rng = Worksheets("Empleados").Range("A3").CurrentRegion
   rng.Sort Key1:=Range("B3"), Order1:=xlAscending, _
            Key2:=Range("C3"), Order2:=xlAscending, _
            Header:=xlYes
End Sub

Private Sub Inicializa_Empleados()
   Dim i As Integer
   '   Inicializa el formulario para la próxima introducción
    With Empleados
        txtNombre = ""
        txtApellido = ""
        optSr = False
        optSra = False
        optSrta = False
        For i = 0 To lstServicios.ListCount - 1
            lstServicios.Selected(i) = False
        Next i
   End With
End Sub

Private Sub lstEmpleados_Click()
   Dim i As Integer
   Dim j As Integer
   '   Muestra el empleado seleccionado
   bNuevo = False
   i = lstEmpleados.ListIndex + 4
   With ThisWorkbook.Worksheets("Empleados")
       Select Case .Cells(i, 1)
           Case "Sr.":    optSr = True
           Case "Sra.":    optSra = True
           Case "Srta.":    optSrta = True
       End Select
       Empleados.txtNombre = .Cells(i, 3)
       Empleados.txtApellido = .Cells(i, 2)
       For j = 0 To Empleados.lstServicios.ListCount - 1
           If Empleados.lstServicios.List(j) = .Cells(i, 4) Then
              Empleados.lstServicios.ListIndex = j
           End If
       Next j
   End With
End Sub
```


Copyrighted Material

Capítulo 7: Mejoras en la interfaz de usuario

A. Presentación 219

B. Ejemplos de barras de comandos 220

1. Barras de herramientas personalizadas 220
2. Grupo "Comandos de menús" en formato Office 2007 . . . 220
3. Agregar un comando a la barra de herramientas de acceso rápido 220

C. Barras de comandos 221

1. Terminología 221
   a. Barra de comandos 221
   b. Control 221
2. Crear una barra de comandos 222
3. Eliminar una barra de comandos 223
4. Mostrar una barra de comandos 224

D. Controles (opciones o botones de comandos) de las barras de comandos 224

1. Agregar un control 224
2. Especificar el título de un control 225
3. Eliminar un control 226

Copyrighted material

4. Asociar un procedimiento a un control 226
5. Otras propiedades 226
6. Lista de imágenes asociadas a los botones de comando . . 228

E. Ejemplos de menús personalizados 230

1. Presentación 230
2. Código de los ejemplos 232
3. Código del módulo de clase ThisWorkbook 232
4. Código de la hoja "Nota de Gastos" 233
5. Código del módulo ProcMenus 233
6. Código del módulo ProcAction 239

Copyrighted Material

A. Presentación

En Microsoft Office 2007, la interfaz de usuario de las versiones anteriores de Excel (formada por barras de herramientas y de menús) se reemplaza por la **Cinta de opciones**, compuesta por diferentes fichas.

De todas formas, la colección **Commandbars** (colección de las barras de herramientas y de menús de Excel) sigue funcionando en Excel 2007. Esto permite acceder al conjunto de barras, paneles de tareas y menús de Excel (barras de comandos, barra de estado, barra de fórmulas, menús contextuales, paneles Office, etc.).

El acceso a esta colección permite realizar las siguientes operaciones con la ayuda del lenguaje VBA:

- crear barras de herramientas personalizadas: estas barras se ubican automáticamente en el grupo **Barras personalizadas** de la ficha **Complementos** (última ficha de la derecha) y su aspecto es el de las barras de herramientas de las versiones anteriores de Excel,
- crear barras de menús personalizadas: se ubican en la ficha **Complementos** o en una hoja Excel bajo la forma de menús contextuales (menús "popup"),
- crear una barra de comandos personalizada en formato de Office 2007,
- personalizar los comandos asignándoles macros.

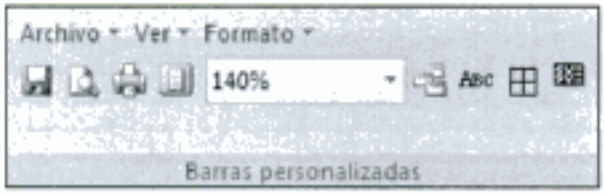
Todas las barras creadas se insertan en la ficha **Complementos** de la cinta de opciones. Es imposible, en la versión 2007, ocultar o reorganizar los comandos de las fichas de la cinta de opciones.

© Editions ENI - All rights reserved

Copyrighted material

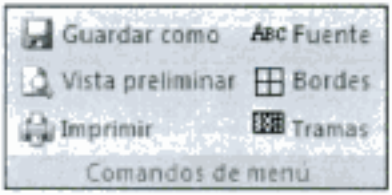
B. Ejemplos de barras de comandos

1. Barras de herramientas personalizadas



Se muestran dos barras de comandos: una barra de herramientas y una barra de menús.

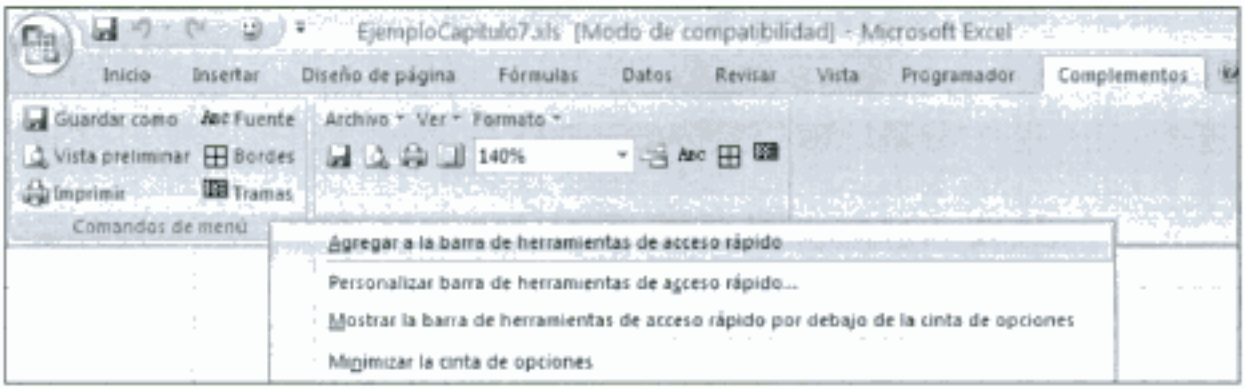
2. Grupo "Comandos de menús" en formato Office 2007



3. Agregar un comando a la barra de herramientas de acceso rápido

Las barras de comandos así creadas pueden hacerse accesibles desde la barra de herramientas de **acceso rápido** de la siguiente manera:

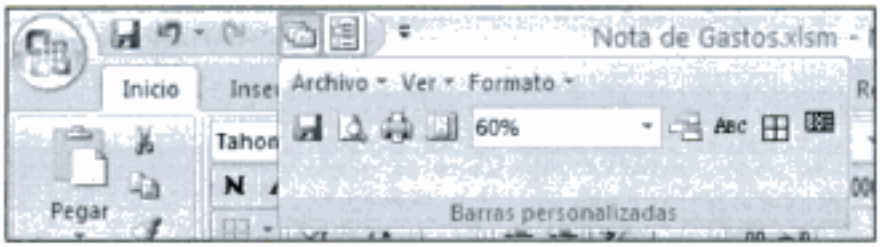
- Ubique el cursor bajo la barra de comandos y haga clic con el botón derecho del ratón,
- Seleccione la opción **Agregar a la barra de herramientas de acceso rápido**.



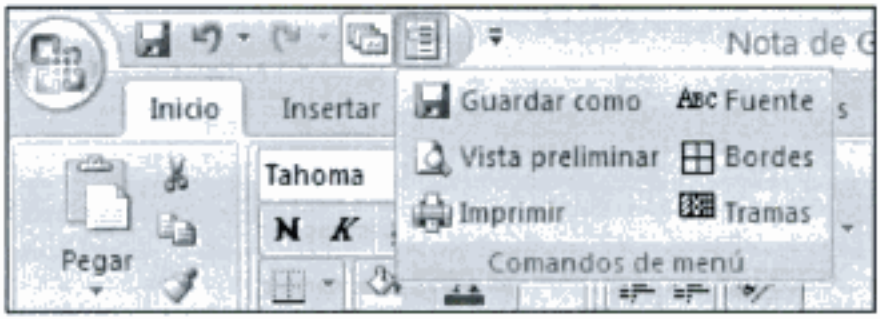
Copyrighted Material

Normalmente, las barras de comandos son accesibles a través de botones de comandos agregados a la barra de herramientas de **acceso rápido**:

Barras de herramientas personalizadas



Comandos de menús



C. Barras de comandos

1. Terminología

a. Barra de comandos

Representa las barras de herramientas de Excel, las barras de herramientas personalizadas y las barras de menús.

b. Control

Representa un comando (botón de comando, opción de menú, etc.) de una barra de comandos.

© Editions ENI - All rights reserved

Copyrighted Material

Hidden page

```
.../...
   '    Crea una barra de herramientas llamada "Menu2"
   Set Barra2 = CommandBars.Add(Name:="Menu2", _
       Position:=msoBarTop)
End Sub
```

Atención: si el código está escrito en el módulo de clase ThisWorkbook, es necesario indicar el objeto **Application** (ejemplo: Application.CommandBars).

3. Eliminar una barra de comandos

`Expression.`**`Delete`**

`Expression`	Expresión que devuelve el objeto **CommandBar** que hay que eliminar.

Ejemplo

Eliminar la barra de menú y la barra de herramientas (indispensable antes de crear nuevamente las barras).

```
Sub Eliminar_Barras()
   'Elimina las barras de comandos personalizadas
   Application.CommandBars("Menu1").Delete
   Application.CommandBars("Menu2").Delete
End Sub
```

Las barras de comandos también pueden referenciarse por el nombre de la variable objeto.

```
Sub Eliminar_Barras()
'   Elimina las barras de comandos personalizadas
      Barra1.Delete
      Barra2.Delete
End Sub
```

© Editions ENI - All rights reserved

Copyrighted material

Hidden page

Hidden page

3. Eliminar un control

`Expression.Delete`

`Expression` Expresión que devuelve el objeto **CommandBarControls** que hay que eliminar.

4. Asociar un procedimiento a un control

→) Use la propiedad **OnAction** del control.

El nombre del procedimiento que desea asociar al control debe indicarse entre comillas.

Para mostrar la tecla de método abreviado del procedimiento asociado, use la propiedad **ShortCutText** del objeto **CommandBarButton**.

---

5. Otras propiedades

→) Para activar o desactivar un control, use la propiedad **Enabled** del control.

→) Para modificar el aspecto de la imagen de un botón, use la propiedad **FaceId** del objeto **CommandBarButton**.

Esta propiedad define el aspecto del botón y no su función.

Ejemplos

Agregar un botón de comando personalizado a la barra de herramientas Menu2. Este botón abre el cuadro de diálogo Guardar como.

```
Sub Agregar_Control1()
    Dim m_Button as CommandBarButton
    '  Agrega un botón de comando a la barra Menu2
     Set m_Button = Application.CommandBars("Menu2").Controls.Add _
           (Type:=msoControlButton)
    '  Icono Guardar
     m_Button.FaceId = 3                                    .../...
```


Copyrighted material

```
.../...
    '  Acción "GuardarComo"
     m_Button.OnAction = "GuardarComo"
End Sub
```

Procedimiento GuardarComo

```
Sub GuardarComo()
    '   Cuadro de diálogo guardar como
     Application.Dialogs(xlDialogSaveAs).Show
End Sub
```

Agregar el menú Archivo y la opción Guardar como a la barra de comandos Menu1.

```
Sub Agrega_Control2()
   Dim m_Menu As CommandBarControl
   Dim m_Option As CommandBarControl
   ' Agrega el menú Archivo
   Set m_Menu = Application.CommandBars("Menu1") _
                .Controls.Add (Type:=msoControlPopup)
   m_Menu.Caption = "Archivo"
   ' Agrega el botón de comando
   Set m_Option = m_Menu.Controls.Add _
   (Type:=msoControlButton)
   m_Option.Caption = "Guardar Como"
   ' Icono Guardar
   m_Option.FaceId = 3
   ' Action "GuardarComo"
   m_Option.OnAction = "GuardarComo"
End Sub
```

© Editions ENI - All rights reserved

Copyrighted material

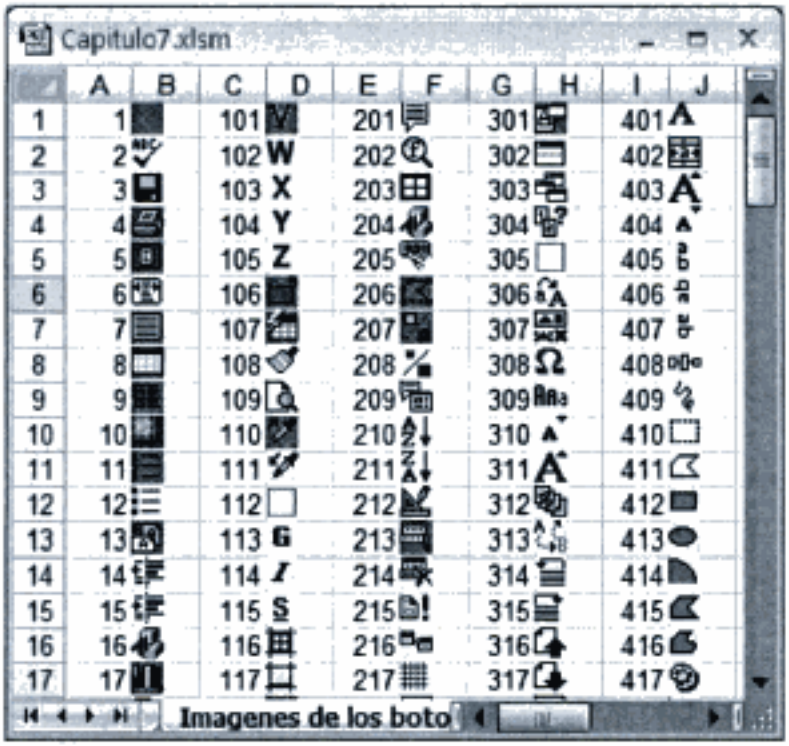
6. Lista de imágenes asociadas a los botones de comando

El siguiente procedimiento muestra, en la hoja Excel activa, la lista de imágenes que se pueden asociar a los botones de comando (propiedad FaceId) y su número correspondiente.

```
Sub Muestra_Imagenes()
   Dim numFila As Integer
   Dim numCol As Integer
   Dim numImagen As Long
   Dim Menu1 As CommandBar
   Dim Button1 As CommandBarControl

   '   Crea una barra de herramientas temporal
   Set Menu1 = Application.CommandBars.Add _
       (Position:=msoBarFloating, temporary:=True)
   '   Agrega un botón de comando
   Set Button1 = Menu1.Controls.Add(msoControlButton)

   '   Modifica la imagen del botón de comando
   '   y la copia en una celda Excel
   For numCol = 1 To 10 Step 2
     For numFila = 1 To 100
           numImagen = numImagen + 1
           Button1.FaceId = numImagen
           Button1.CopyFace
           ActiveSheet.Cells(numFila, numCol) = numImagen
           ActiveSheet.Paste Cells(numFila, numCol + 1)
     Next numFila
   Next numCol

   '   Redimensiona las columnas
   Columns("A:W").Select
   Selection.ColumnWidth = 4
   '   Elimina la barra de herramientas
   Menu1.Delete

End Sub
```


Copyrighted Material

El resultado obtenido es el siguiente:

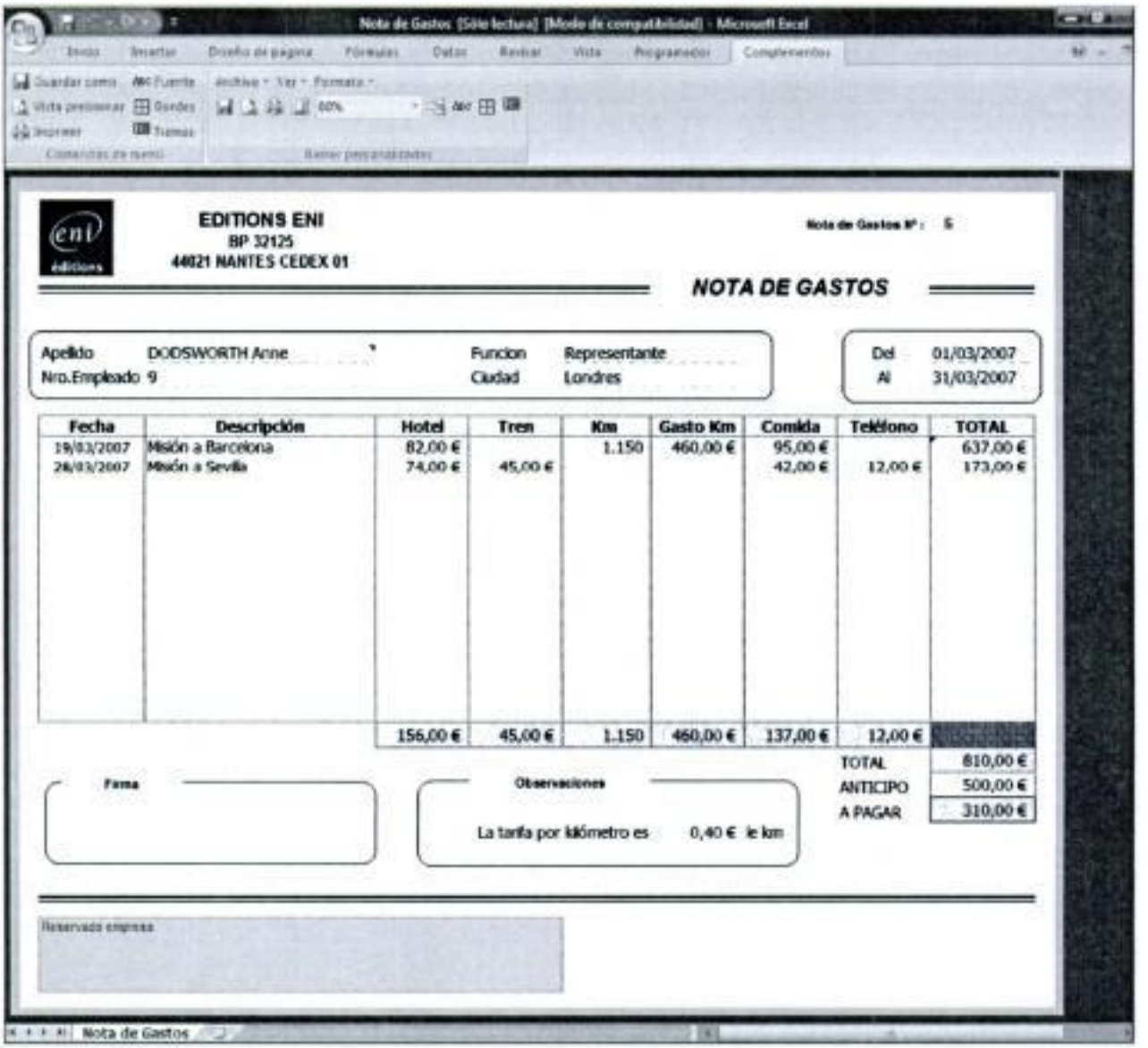


© Editions ENI - All rights reserved

Copyrighted material

E. Ejemplos de menús personalizados

1. Presentación



EDITIONS ENI
BP 32125
44021 NANTES CEDEX 01

Nota de Gastos Nº : 5

NOTA DE GASTOS

Apellido DODSWORTH Anne
Nro.Empleado 9
Funcion Representante
Ciudad Londres
Del 01/03/2007
Al 31/03/2007

Fecha	Descripción	Hotel	Tren	Km	Gasto Km	Comida	Teléfono	TOTAL
19/03/2007	Misión a Barcelona	82,00 €		1.150	460,00 €	95,00 €		637,00 €
26/03/2007	Misión a Sevilla	74,00 €	45,00 €			42,00 €	12,00 €	173,00 €
		156,00 €	45,00 €	1.150	460,00 €	137,00 €	12,00 €	

TOTAL 810,00 €
ANTICIPO 500,00 €
A PAGAR 310,00 €

Firma

Observaciones

La tarifa por kilómetro es 0,40 € le km

Reservado empresa

Nota de Gastos

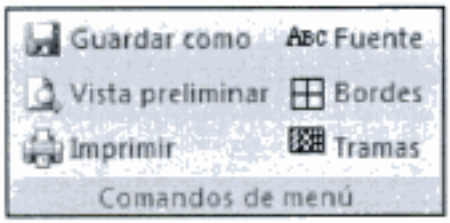
Copyrighted Material

Este ejemplo crea las siguientes barras de comandos:

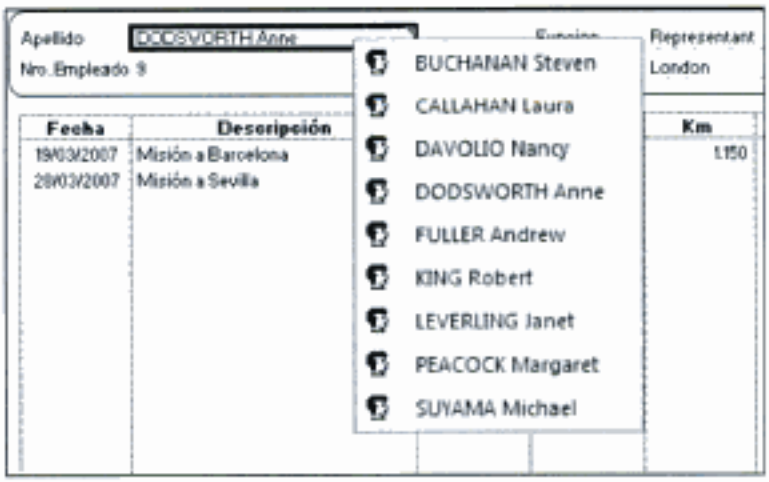
- Una barra de menú con el título "Menú Gastos" que permite acceder a las siguientes opciones:

- Una barra de menú con el título "Gastos":

- Un grupo de comandos con el título "Comandos de menú":



- Una barra de menú contextual que aparece cuando el usuario se posiciona en la zona llamada "Empleado" y hace clic con el botón secundario del ratón. Los empleados se extraen de la base Access Empleados.accdb (base disponible con los ejemplos del libro).



© Editions ENI - All rights reserved

Copyrighted material

2. Código de los ejemplos

Para crear los menús del siguiente ejemplo, se deben realizar las siguientes operaciones:

- crear el documento Excel;
- definir un área de impresión;
- definir un rango de celdas con el nombre "Empleado" que incluya las celdas Apellido y el número de empleado;
- definir un rango de celdas con el nombre "NotaDeGastos" que incluya las celdas que hay que imprimir;
- asignar los siguientes nombres a las celdas que contienen la información del empleado: apellido, númemp, función, ciudad;
- agregar un módulo estándar llamado **ProcMenus**; este módulo contendrá los procedimientos para crear las diferentes barras de comandos;
- agregar un módulo estándar llamado **ProcAcciones**; este módulo contendrá los procedimientos personalizados asociados a los botones de comandos.

3. Código del módulo de clase ThisWorkbook

```
option Explicit
_____________________________________________
Private Sub Workbook_Open()
   '   Muestra los menús personalizados
   Personalizar_Excel
   '   Ajusta el zoom
   Ajuste

End Sub
_____________________________________________
Private Sub Workbook_BeforeClose(Cancel As Boolean)
   '   Pide confirmación del cierre del libro
   If MsgBox("¿Desea cerrar el libro?", _
       vbQuestion & vbYesNo, "Nota de Gastos") = vbYes Then
      ' Muestra los menús de Excel
      Restaurar_Excel
   Else
      Cancel = True
   End If
End Sub
```


Copyrighted material

4. Código de la hoja "Nota de Gastos"

```
Private Sub Worksheet_BeforeRightClick(ByVal Target As Excel.Range, _
            Cancel As Boolean)
   '   Si la primera celda activa pertenece al rango llamado
   '   Empleado: mostrar el menú Empleados
   If Union(Target.Range("A1"), Range("Empleado")).Address = _
      Range("Empleado").Address Then
      CommandBars("Empleados").ShowPopup
      Cancel = True
End If
End Sub
```

5. Código del módulo ProcMenus

```
option Explicit

'   Declaración de variables
Dim m_Menu As CommandBarPopup
Dim m_Barra As CommandBar
Dim m_Option As CommandBarControl
Dim m_Button As CommandBarButton

Sub Personalizar_Excel()
   Dim i As Integer
   '   Crea los menús y comandos personalizados
   Mostrar_Barra_Menús
   Mostrar_Barra_Herramientas
   Mostrar_Barra_Comandos
   '   Menú contextual Lista de empleados
   Lista_Empleados
   '   Oculta las barras de fórmulas, de estado
   '   y los encabezados de fila y columna
   Application.DisplayFormulaBar = False
   Application.DisplayStatusBar = False
   Application.ActiveWindow.DisplayHeadings = False
End Sub
_____________________________________________________________
Sub Restaurar_Excel()
   Dim i As Integer
   Dim m_Ctrl As CommandBarControl
   '   Elimina los menús y barras de comandos
   On Error Resume Next
   Application.CommandBars("Menú Gastos").Delete                .../...
```

© Editions ENI - All rights reserved

Copyrighted material

```
.../...
   Application.CommandBars("Gastos").Delete
   Application.CommandBars("Empleados").Delete
   On Error GoTo 0
   '    Elimina los comandos personalizados
   '    de la barra de comando Herramientas
   For Each m_Ctrl In Application.CommandBars("Tools").Controls
       If Not m_Ctrl.BuiltIn Then
          m_Ctrl.Delete
       End If
   Next m_Ctrl
   '    Vuelve a mostrar las barras de fórmulas, de estado
   '    y los encabezados de fila y columna
   Application.DisplayFormulaBar = True
   Application.DisplayStatusBar = True
   Application.ActiveWindow.DisplayHeadings = True
   End Sub
   Sub Mostrar_Barra_Menus()
   On Error Resume Next
   '     Elimina la barra de Menú para recrearla
   Application.CommandBars("Menu Gastos").Delete
   On Error GoTo 0
   '    Crea la barra de menú
   Set m_Barra = Application.CommandBars.Add(Name:="Menu Gastos", _
       Position:=msoBarTop)
   '    Muestra la barra de menú creada
   Application.CommandBars("Menu Gastos").Visible = True
   '    Agrega el menú Archivo
   Set m_Menu = m_Barra.Controls.Add(Type:=msoControlPopup)
   m_Menu.Caption = "Archivo"
   '    Agrega los comandos del menú Archivo
   '    Las acciones son las acciones por defecto
   Set m_Option = m_Menu.Controls.Add(Type:=msoControlButton, ID:=3)
   m_Option.OnAction = "GuardarComo"
   Set m_Option = m_Menu.Controls.Add(Type:=msoControlButton, ID:=109)
   Set m_Option = m_Menu.Controls.Add(Type:=msoControlButton, ID:=4)
   Set m_Option = m_Menu.Controls.Add(Type:=msoControlButton, ID:=106)
   '    Agrega el menú "Ver"
   Set m_Menu = m_Barra.Controls.Add(Type:=msoControlPopup)
   m_Menu.Caption = "Ver"
   '    Agrega la lista desplegable de zoom
   Set m_Option = m_Menu.Controls.Add(Type:=msoControlComboBox, ID:=1733)
   '    Agrega los comandos del menú Ver
   '    con el llamado a los procedimientos acción
   Set m_Option = m_Menu.Controls.Add(Type:=msoControlButton)
   m_Option.FaceId = 175                                          .../...
```


Copyrighted Material

Hidden page

Hidden page

```
.../...
      .TooltipText = "Tramas"
   End With
End Sub
```

```
Sub Mostrar_Barra_Comandos()
   '    Objeto que representa la barra de herramientas "Tools"
   Set m_Barra = Application.CommandBars("Tools")
   '    Agrega los botones de comandos estándares en la barra de
        herramientas
   Set m_Button = m_Barra.Controls.Add _
       (Type:=msoControlButton, ID:=3)
   m_Button.Caption = "Guardar como"
   m_Button.OnAction = "GuardarComo"
   Set m_Button = m_Barra.Controls.Add _
       (Type:=msoControlButton, ID:=109)
   m_Button.Caption = "Vista preliminar"
   m_Button.OnAction = "VistaPreliminar"
   Set m_Button = m_Barra.Controls.Add _
       (Type:=msoControlButton, ID:=4)
   m_Button.Caption = "Imprimir"
   m_Button.OnAction = "Imprimir"
   '    Agrega los comandos de tipo Formato
   Set m_Button = m_Barra.Controls.Add _
       (Type:=msoControlButton)
   With m_Button
      .FaceId = 291
      .OnAction = "Fuente"
      .TooltipText = "Fuente"
      .Caption = "Fuente"
   End With
   Set m_Button = m_Barra.Controls.Add _
       (Type:=msoControlButton)
   With m_Button
      .FaceId = 1704
      .OnAction = "Bordes"
      .TooltipText = "Bordes"
      .Caption = "Bordes"
   End With
   Set m_Button = m_Barra.Controls.Add _
       (Type:=msoControlButton)
   With m_Button
      .FaceId = 1988
      .OnAction = "Tramas"
      .TooltipText = "Tramas"                                    .../...
```

© Editions ENI - All rights reserved

Copyrighted material

Hidden page

6. Código del módulo ProcAction

```
Sub Ajuste()
   '   Ajusta el zoom al contenido
   '   del rango llamado "NotaDeGastos"
   Application.Goto Reference:="NotaDeGastos"
   ActiveWindow.Zoom = True
   Range("Apellido").Select
End Sub
```

```
Sub Mostrar_Empleado(Numemp As Long)
   Dim Db As Database
   Dim rstEmp As Recordset
   Dim strSql As String
   '   Abre la tabla Empleados
   Set Db = OpenDatabase(ActiveWorkbook.Path & "\Empleados.accdb")
   strSql = "SELECT * FROM Empleados WHERE [Nº de empleados] = " _
            & Numemp
   Set rstEmp = Db.OpenRecordset(strSql)
   '   Muestra las coordenadas del empleado seleccionado
   Range("Apellido") = UCase(rstEmp("Apellido")) & " " & _
                            rstEmp("Nombre")
   Range("Numemp") = rstEmp("Nº de empleados")
   Range("Funcion") = rstEmp("Funcion")
   Range("Ciudad") = rstEmp("Ciudad")
   '   Cierra los objetos Access
   rstEmp.Close
   Db.Close
End Sub
```

```
Sub Fuente()
   '   Muestra el cuadro de diálogo Fuente
   Application.Dialogs(xlDialogFormatFont).Show
End Sub
```

```
Sub Bordes()
   '   Muestra el cuadro de diálogo Bordes
   Application.Dialogs(xlDialogBorder).Show
End Sub
```

```
Sub Tramas()
   '   Muestra el cuadro de diálogo Tramas
   Application.Dialogs(xlDialogPatterns).Show
End Sub
```

.../...

© Editions ENI - All rights reserved

Copyrighted Material

Hidden page

Capítulo 8: Administración de eventos

A. Presentación 242

B. Escritura de eventos 242

1. Eventos de libro, de hoja o de formulario 242
2. Eventos del objeto Application 244
3. Evento asociado a un gráfico incrustado 247

C. Eventos del objeto Application 250

D. Eventos del objeto Workbook 253

E. Eventos del objeto Worksheet 256

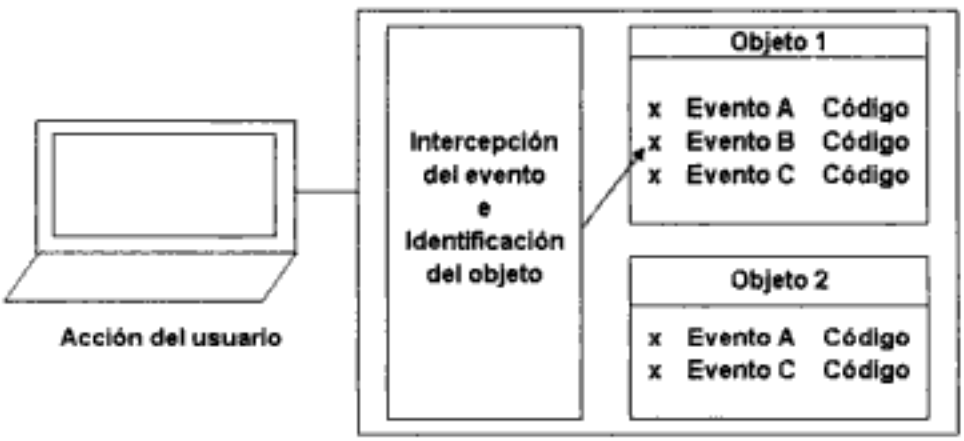
F. Eventos del objeto Chart 257

Copyrighted material

A. Presentación

Un evento es una **acción** del usuario o del sistema reconocido por un objeto de Microsoft Excel. El evento desencadena un procedimiento asociado al evento del objeto activo.

Estos procedimientos le permiten asociar un código personalizado en respuesta a un evento que se produce en un objeto Excel (libro, hoja, formulario, gráfico, etc.).



B. Escritura de eventos

1. Eventos de libro, de hoja o de formulario

Usted puede acceder a los procedimientos de eventos asociados a un objeto de la siguiente manera:

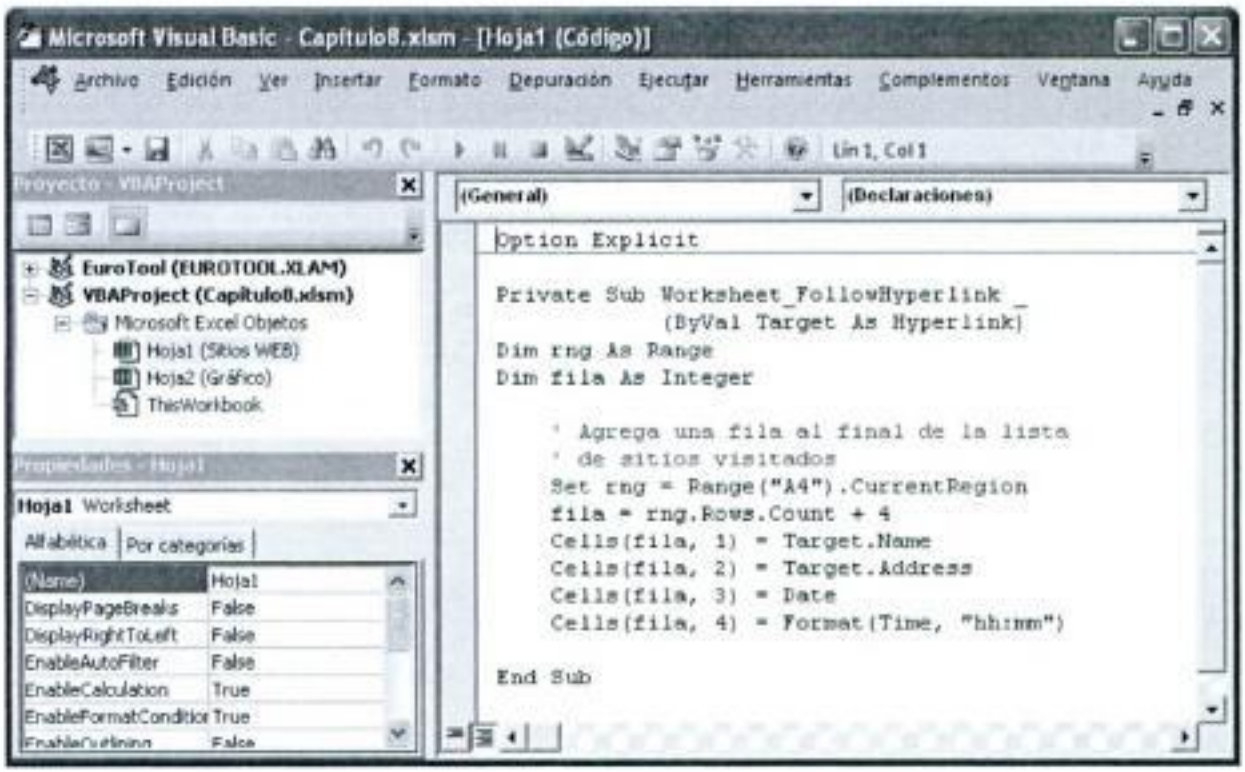
- En la ventana **Explorador de proyectos**, haga doble clic en el objeto deseado (libro, hoja o formulario) para hacer aparecer la ventana de código correspondiente.
- Abra la lista desplegable a la izquierda de la ventana de código y seleccione **Workbook**, **Worksheet** o **UserForm**, según el objeto seleccionado.
- También puede seleccionar un evento vinculado al objeto seleccionado en la lista desplegable de la derecha para asociarle un código personalizado.

Copyrighted Material

La ejecución de los procedimientos de eventos puede desactivarse en cualquier momento asignando el valor **False** a la propiedad **EnableEvents** del objeto **Application**.

Ejemplo

Este ejemplo muestra cómo obtener un listado histórico de todos los hipervínculos visitados en la hoja de cálculo activa.



© Editions ENI - All rights reserved

Copyrighted material

Resultado en Excel:

Capitulo8.xlsm [Modo de compatibilidad]

	A	B	C	D	E	F	G
1							
2		**SITIOS VISITADOS**				**SITIOS DISPONIBLES**	
3							
4	**Nombre**	**Dirección**	**Fecha**	**Hora**			
5	Microsoft	http://www.microsoft.com	11/07/2007	14:41		Microsoft	
6	RENFE	http://www.renfe.es	12/07/2007	4:36		Ediciones ENI	
7	Iberia	http://www.iberia.es	19/07/2007	5:51		Bolsa de Madrid	
8	Bolsa de Madrid	http://www.bolsamadrid.es	21/07/2007	1:11		El tiempo	
9	El tiempo	http://www.meteofa.mil.ar	28/07/2007	4:45		RENFE	
10	RENFE	http://www.renfe.es	29/07/2007	7:47		Iberia	
11	Microsoft	http://www.microsoft.com	31/07/2007	22:07			
12							
13							

Sitios WEB | Gráfico

2. Eventos del objeto Application

Se necesitan tres etapas para la escritura y ejecución de los eventos del objeto **Application**.

Etapa 1

- Inserte un módulo de clase:

  Insertar
  Módulo de clase

  o abra la lista y haga clic en **Módulo de clase**.

- Una vez insertado el módulo, asígnele un nombre.

Ejemplo

*Dele el nombre **ObjApplication** al módulo de clase.*

Etapa 2

- En el módulo de clase, cree un objeto **Application** con el siguiente código:

```
Public WithEvents NomObjeto As Application
```


Copyright material

Ejemplo

*Creación del objeto **MiAplicacion** como aplicación.*

```
Public WithEvents MiAplicacion As Application
```

El objeto así creado queda disponible en la lista de la izquierda del módulo.

→ Seleccione el objeto creado en la lista de la izquierda del módulo y luego seleccione el evento esperado en la lista de la derecha. Escriba el código de los procedimientos que desea generar.

Ejemplo

Creación de dos procedimientos de eventos: el primero realiza la inserción de una nueva hoja; el segundo, la creación de un nuevo libro.

```
Public WithEvents MiAplicacion As Excel.Application
_____________________________________________________________
Private Sub MiAplicacion_WorkbookNewSheet _
            (ByVal Wb As Workbook, ByVal Sh As Object)
   Dim NomHoja As String
   '  Cada vez que se agrega una hoja se pide al usuario
   '  que introduzca un nombre que a continuación se destinará a la hoja
   '  insertada tras las horas existentes
   NomHoja = InputBox("Introduzca el nombre de la hoja")
   ActiveSheet.Name = NomHoja
   ActiveSheet.Move After:=Sheets(Sheets.Count)
End Sub
_____________________________________________________________
Private Sub MiAplicacion_NewWorkbook(ByVal Wb As Workbook)
   Dim NbHojas As Integer
   Dim NbActual As Integer
   Dim Diferencia As Integer
   '  Por cada nuevo libro,
   '  solicitamos al usuario la cantidad de hojas
   '  Según el caso, se agregan o eliminan las hojas necesarias
   Do
      NbHojas = Application.InputBox _
                  ("¿Cantidad de hojas?", Type:=1)
   Loop While NbHojas = False
   NbActual = Sheets.Count
   Diferencia = NbActual - NbHojas                          .../...
```

© Editions ENI - All rights reserved

Copyrighted material

```
.../...
   '   Eliminar las hojas de más
   '   Supresión de los mensajes de alerta con el fin
   '   de no obtener mensajes en la supresión de hojas
   Do While Diferencia > 0
      Application.DisplayAlerts = False
      Sheets.Item(Diferencia).Select
      ActiveWindow.SelectedSheets.Delete
      Diferencia = Diferencia - 1
   Loop

   '   Agregar hojas necesarias
   '   Se desactivan los eventos para
   '   no indicar los nombres de las nuevas hojas
   Do While Diferencia < 0
      Application.EnableEvents = False
      Sheets.Add
       Diferencia =  Diferencia + 1
   Loop
   '   Reactivar eventos y alertas
   Application.EnableEvents = True
   Application.DisplayAlerts = True
End Sub
```

Etapa 3

→) Active un módulo cualquiera y conecte el objeto declarado en el módulo de clase con el objeto **Application** para las siguientes instrucciones:

```
Dim NomVariable As New NomModuloDeClase

Sub NomProced ()
Set NomVariable.NomObjeto = Application
End Sub
```


Copyrighted Material

Hidden page

Ejemplo

*Dele el nombre **ObjGraficos*** al módulo de clase.

Etapa 2

→ En el módulo de clase, cree un objeto gráfico para el siguiente código:

```
Public WithEvents NomObjeto As Chart
```

Ejemplo

*Creación del objeto llamado **Chart1** como gráfico incrustado.*

```
Public WithEvents Chart1 As Chart
```

El objeto así creado queda disponible en la lista de la izquierda del módulo.

→ Seleccione el objeto creado en la lista de la izquierda del módulo y luego seleccione el evento esperado en la lista de la derecha. Escriba el código de los procedimientos que desea generar.

Ejemplo

Creación de dos procedimientos de eventos: uno desactiva el gráfico, el otro especifica sus dimensiones.

```
Option Explicit
Public WithEvents Chart1 As Chart
_____________________________________________________________
Private Sub Chart1_Deactivate()
   Dim Respuesta As String
   '  Cada vez que se desactiva el gráfico
   '  se pregunta si hay que guardar el libro
   Respuesta = MsgBox("¿Guardar los cambios?", vbYesNo)
   If Respuesta = vbYes Then ActiveWorkbook.Save
End Sub                                                  .../...
```


Copyrighted material

Hidden page

Finalmente, llame al procedimiento InitMiAplicacion al abrir el libro (módulo de clase ThisWorkbook).

```
Private Sub Workbook_Open()
InicializaGrafico
End Sub
```

Al abrir este libro, los procedimientos creados durante la etapa 2 se ejecutarán automáticamente y se redimensionará o se desactivará el gráfico situado en la segunda hoja de cálculo. Estos procedimientos se desactivarán al cerrar el libro.

C. Eventos del objeto Application

CalculateUntilAsyncQueriesDone

Ejecuta todas las consultas en curso en las fuentes de datos OLEDB y OLAP.

NewWorkBook

Ocurre al crear un nuevo libro.

SheetActivate

Ocurre al activar una hoja.

SheetBeforeDoubleClick

Ocurre al hacer doble clic en una hoja de cálculo, antes de la acción predeterminada para el doble clic.

SheetBeforeRightClick

Ocurre al hacer clic con el botón secundario del ratón en una hoja de cálculo, antes de la acción predeterminada.

SheetCalculate

Ocurre cuando se recalcula toda la hoja de cálculo o después de que se actualiza un gráfico al modificar sus datos.

Copyrighted material

SheetChange

Ocurre cuando las celdas de una hoja de cálculo se modifican por el usuario o por un vínculo externo.

SheetDeactivate

Ocurre al desactivar una hoja de cálculo.

SheetFollowHyperlink

Ocurre cuando el usuario hace clic en un hipervínculo en Microsoft Excel.

SheetPivotTableUpdate

Ocurre al actualizar la hoja de informe de una tabla dinámica.

SheetSelectionChange

Ocurre cuando cambia la selección en cualquier hoja de cálculo (el evento no ocurre si la selección se hace sobre una hoja de gráfico).

WindowActivate

Ocurre al activar una ventana de libro.

WindowDeActivate

Ocurre cuando se desactiva una ventana de libro.

WindowResize

Ocurre al cambiar el tamaño de una ventana de libro.

WorkBookActivate

Ocurre cuando se activa un libro.

WorkBookAddinInstall

Ocurre cuando se instala un libro bajo la forma de una macro complementaria.

WorkBookAddinUninstall

Ocurre cuando se desinstala una macro complementaria.

WorkbookAfterXMLExport

Ocurre después de exportar un archivo XML.

© Editions ENI - All rights reserved

Copyrighted material

WorkbookAfterXMLImport

Ocurre después de importar un archivo XML.

WorkBookBeforeClose

Ocurre justo antes de cerrar un libro.

WorkBookBeforePrint

Ocurre antes de imprimir un libro abierto.

WorkBookBeforeSave

Ocurre antes de guardar un libro abierto.

WorkbookBeforeXMLExport

Ocurre antes de exportar un archivo XML.

WorkbookBeforeXMLImport

Ocurre antes de importar un archivo XML.

WorkBookDeactivate

Ocurre cuando se desactiva un libro abierto.

WorkBookNewSheet

Ocurre cuando se crea una nueva hoja en un libro abierto.

WorkBookOpen

Ocurre cuando se abre un libro.

WorkbookPivotTableOpenConnection

Ocurre al abrir la conexión de un informe de tabla dinámica con su fuente de datos.

WorkbookPivotTableCloseConnection

Ocurre al cerrar la conexión de un informe de tabla dinámica con su fuente de datos.

WorkbookRowsetComplete

Ocurre cuando el usuario extrae el juego de grabación de una tabla dinámica OLAP.

Copyrighted Material

WorkbookSync

Ocurre al sincronizar la copia local de una hoja de cálculo hecha a partir de un área de trabajo con la copia en el servidor.

D. Eventos del objeto Workbook

Activate

Ocurre cuando se activa el libro.

AddinInstall

Ocurre cuando el libro se instala bajo la forma de una macro complementaria.

AddinUninstall

Ocurre cuando el libro se desinstala bajo la forma de una macro complementaria.

AfterXMLExport

Ocurre después de exportar un archivo XML.

AfterXMLImport

Ocurre después de importar un archivo XML.

BeforeClose

Ocurre antes de cerrar el libro; si el libro fue modificado, este evento se produce antes de invitar al usuario a guardar los cambios.

BeforePrint

Ocurre antes de imprimir el libro (o cualquiera de sus partes).

BeforeSave

Ocurre antes de grabar el libro.

BeforeXMLExport

Ocurre antes de exportar un archivo XML.

© Editions ENI - All rights reserved

Copyrighted material

BeforeXMLImport

Ocurre antes de importar un archivo XML.

Deactivate

Ocurre al desactivar un gráfico, una hoja de cálculo o un libro.

NewSheet

Ocurre cuando se crea una nueva hoja en el libro.

Open

Ocurre cuando se abre el libro.

PivotTableOpenConnection

Ocurre al abrir la conexión de un informe de tabla dinámica con su fuente de datos.

PivotTableCloseConnection

Ocurre al cerrar la conexión de un informe de tabla dinámica con su fuente de datos.

RowsetComplete

Este evento se desencadena cuando el usuario extrae el juego de grabación de una tabla dinámica OLAP.

SheetActivate

Ocurre cuando se activa una hoja.

SheetBeforeDoubleClick

Ocurre al hacer doble clic en una hoja de cálculo, antes de la acción predeterminada para el doble clic.

SheetBeforeRightClick

Ocurre al hacer clic con el botón secundario del ratón en una hoja de cálculo, antes de la acción predeterminada.

SheetCalculate

Ocurre cuando se recalcula toda la hoja de cálculo o después de que se recalcula un gráfico al modificar sus datos.

Copyrighted material

SheetChange

Ocurre cuando las celdas de una hoja de cálculo se modifican por el usuario o por un vínculo externo.

SheetDeactivate

Ocurre cuando se desactiva una hoja de cálculo.

SheetFollowHyperlink

Se produce al hacer clic en cualquier hipervínculo en Microsoft Excel.

SheetPivotTableUpdate

Ocurre al actualizar la hoja del informe de tabla dinámica.

SheetSelectionChange

Se produce cuando cambia la selección en una hoja de cálculo cualquiera (el evento no se produce si la selección está en una hoja de gráfico).

Sync

Ocurre al sincronizar la copia local de una hoja de cálculo hecha a partir de un área de trabajo con la copia en el servidor.

WindowActivate

Ocurre al activar un libro.

WindowDeActivate

Ocurre al desactivar un libro.

WindowResize

Ocurre cuando cambia el tamaño de la ventana.

© Editions ENI - All rights reserved

Copyrighted Material

E. Eventos del objeto Worksheet

Activate

Ocurre cuando se activa un libro, una hoja de cálculo, una hoja de gráfico o un gráfico incrustado.

BeforeDoubleClick

Ocurre al hacer doble clic en una hoja de cálculo o un gráfico incrustado, antes de la acción predeterminada para el doble clic.

BeforeRightClick

Ocurre al hacer clic con el botón secundario del ratón en una hoja de cálculo o un gráfico incrustado, antes de la acción predeterminada.

Calculate

Ocurre al recalcular la hoja de cálculo.

Change

Ocurre cuando algunas celdas de la hoja de cálculo están modificadas por el usuario o por un vínculo externo.

Deactivate

Ocurre al desactivar el gráfico, la hoja de cálculo o el libro.

FollowHyperlink

Ocurre al hacer clic en un hipervínculo de una hoja de cálculo.

PivotTableUpdate

Ocurre después de actualizar un informe de tabla dinámica en una hoja de cálculo.

SelectionChange

Ocurre cuando cambia la selección en una hoja de cálculo.

Copyrighted Material

F. Eventos del objeto Chart

Activate

Ocurre cuando se activa una hoja de gráfico o un gráfico incrustado.

BeforeDoubleClick

Ocurre al hacer doble clic en un gráfico incrustado o una hoja de gráfico, antes de la acción predeterminada para el doble clic.

BeforeRightClick

Ocurre al hacer clic con el botón secundario en un gráfico incrustado o una hoja de gráfico, antes de la acción predeterminada correspondiente.

Calculate

Ocurre después de que el gráfico se actualice con datos nuevos o modificados.

Deactivate

Ocurre cuando se desactiva el gráfico, la hoja de cálculo o el libro.

DragOver

Ocurre al arrastrar un rango de celdas sobre un gráfico.

DragPlot

Ocurre al arrastrar y colocar un rango de celdas sobre un gráfico.

MouseDown

Ocurre al presionar el botón izquierdo o derecho del ratón cuando el puntero está sobre un gráfico.

MouseMove

Ocurre al cambiar la posición del puntero del ratón sobre un gráfico.

© Editions ENI - All rights reserved

Copyrighted material

MouseUp

Ocurre al soltar el botón izquierdo o derecho del ratón cuando el puntero está sobre un gráfico.

Resize

Se produce al cambiar el tamaño del gráfico.

Select

Se produce al seleccionar un elemento del gráfico.

SeriesChange

Ocurre cuando el usuario modifica el valor de un punto de datos del gráfico.

Copyrighted material

Capítulo 9: Depuración y administración de errores

A. Diferentes tipos de error 260

1. Errores de sintaxis 260
2. Errores de compilación 261
3. Errores de ejecución 262
4. Errores de lógica 262

B. Depuración 263

1. Presentación 263
2. La barra de herramientas Depuración 263
3. El objeto Debug 264

C. Administración de errores en VBA 265

1. El objeto Err 268

Copyrighted material

A. Diferentes tipos de error

Se distinguen diferentes tipos de error en el lenguaje VBA:

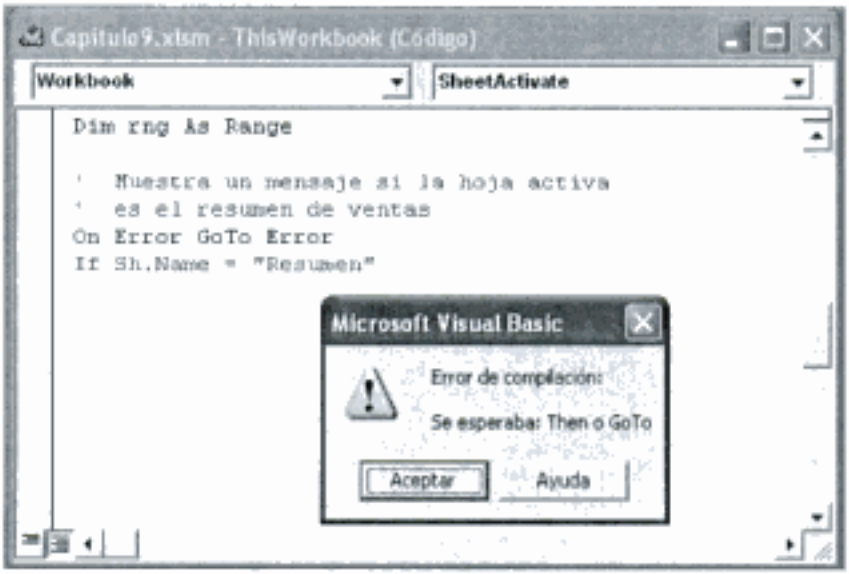
- errores de sintaxis;
- errores de compilación;
- errores de ejecución;
- errores de lógica.

1. Errores de sintaxis

Los errores de sintaxis se detectan automáticamente a medida que se introduce el código en VBA.

→ Para activar la comprobación de sintaxis, en el menú **Herramientas**, seleccione **Opciones**, luego seleccione la ficha **Editor** y marque la casilla **Comprobación de sintaxis automática**.

Ejemplo



Los errores de sintaxis no corregidos provocarán un error de compilación, de ahí el mensaje que aparece.

Copyrighted Material

2. Errores de compilación

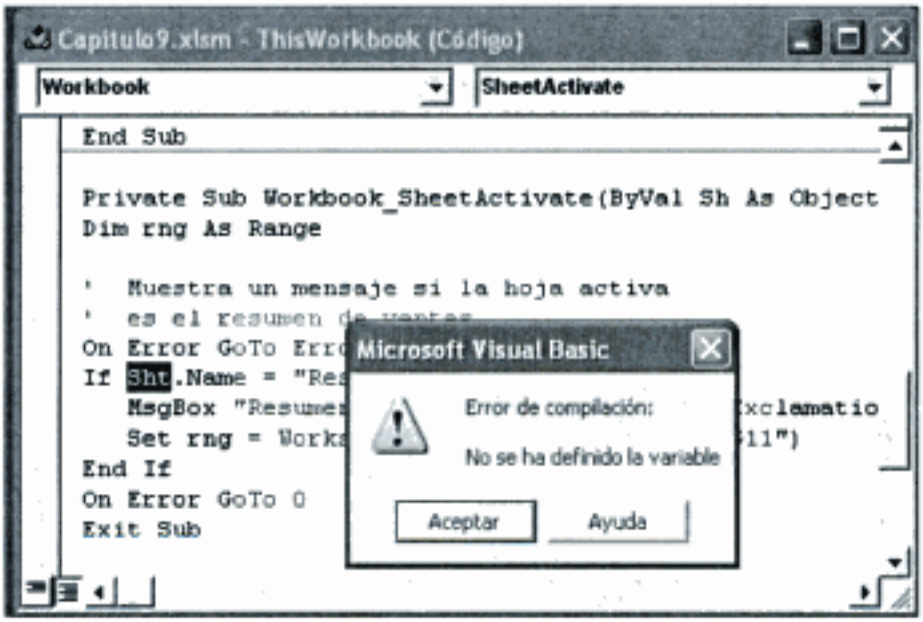
Los errores de compilación se detectan cuando Excel trata de compilar el código.

El código VBA puede compilarse de dos maneras:

- a pedido, al seleccionar la opción **Compilar VBAProject** del menú **Depuración**. En este caso, el código se compila en su totalidad.
- automáticamente **al ejecutar el código**. En este caso, solamente se compila el código contenido en los procedimientos cuando son llamados por primera vez. Los procedimientos no llamados no se compilarán.

Se recomienda compilar el programa antes de ejecutarlo para ganar tiempo en la actualización.

Ejemplo



Es posible anticipar los errores de ejecución debidos a las variables no declaradas usando la instrucción **Option Explicit**. Si trata de usar un nombre de variable no declarado, se produce un error durante la compilación.

© Editions ENI - All rights reserved

Copyrighted material

3. Errores de ejecución

Los errores de ejecución se detectan cuando Excel trata de ejecutar el código. Una instrucción, una operación, una llamada a una función, etc. inválidas provocan un error de ejecución. Por ejemplo, el uso de un índice erróneo en una colección o la asignación de un valor no numérico a una variable numérica pueden provocar un error de ejecución.

Ejemplo

La hoja de cálculo "Enero" no existe en el libro activo.

4. Errores de lógica

Los errores de lógica tienen que ver con errores de razonamiento o con una mala traducción de un razonamiento en código VBA. Por ejemplo, un algoritmo de cálculo puede producir un error en el resultado si en su transcripción a VBA, si se omite o se traduce mal una operación o si el algoritmo es erróneo.

Los errores de lógica son los más difíciles de detectar. En general, no producen un error de ejecución: pero producen un resultado distinto del esperado.

Copyright © RA-MA Material

Hidden page

Hidden page

Métodos

Print

Muestra texto en la ventana **Inmediato**.

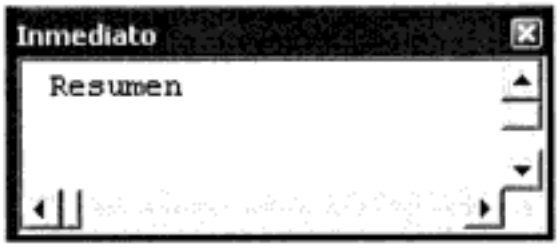
Assert

Suspende de forma condicional la ejecución de la línea en la que aparece el método.

Ejemplo

```
Private Sub Workbook_SheetActivate(ByVal Sh As Object)
'   Muestra el nombre de la hoja activa en la ventana Inmediato
Debug.Print Sh.Name
If Sh.Name = "Resumen" Then ...
```

Resultado en la ventana Ejecución:



C. Administración de errores en VBA

Cuando se produce un error, VBA genera un error de ejecución que interrumpe la aplicación. Otros errores pueden hacer que el código VBA se comporte de manera imprevisible.

Para evitar esto, es posible manejar el error con la ayuda de las siguientes instrucciones y funciones:

➔ On Error

(instrucción)

Indica una secuencia de instrucciones que se ejecutará en caso de error.

© Editions ENI - All rights reserved

Copyrighted material

Sintaxis 1

```
On Error GoTo línea
```

Activa la rutina de administración de errores que comienza en la línea indicada por el argumento `línea`.
El argumento `línea` debe ser una etiqueta o número de línea.
La `línea` debe pertenecer al mismo procedimiento que la instrucción **On Error**.
Si el argumento `línea` es un número de línea, debe ser obligatoriamente el primer caracter no vacío de la línea.

Sintaxis de la rutina de administración de errores

```
Línea:
   instrucciones
Resume
```

La instrucción **Resume** permite continuar la ejecución del código cuando termina la rutina de administración de errores, es decir, una vez resuelto el problema que produjo el error.

Hay tres sintaxis diferentes para **Resume**:

Resume 0 reanuda la ejecución del código donde el error se produjo.

Resume Next reanuda a partir de la instrucción que sigue inmediatamente a la que generó el error.

Resume Línea reanuda en la línea especificada por el argumento `Línea`.

Para impedir la ejecución del código de administración de errores cuando no hay errores, coloque una instrucción **Exit Sub**, **Exit Function** o **Exit Property** inmediatamente antes de la rutina de administración de errores.

Sintaxis 2

```
On Error Resume Next
```

Especifica que, en caso de error, la ejecución debe continuar.

Copyrighted Material

Sintaxis 3

`On Error GoTo` 0

Permite interrumpir la administración de errores cuando el procedimiento está todavía en ejecución.

Ejemplo

Este procedimiento selecciona cada hoja y le cambia el nombre (por medio de un cuadro de diálogo) con una rutina de administración de error que se ejecuta cuando el nombre elegido es incorrecto o corresponde a un nombre existente.

```
Sub Errores_Nombre_Hojas()
   Dim HojaTest As Worksheet
   Dim NuevoNombre As String

   '  En caso de error, se ejecutará la rutina "AdministracionDeErrores"
   On Error GoTo AdministracionDeErrores

   '  Para cada hoja, seleccionarla y solicitar un nombre
   For Each HojaTest In Sheets
       HojaTest.Select
   1   NuevoNombre = InputBox _
               (prompt:="Escriba el nombre de hoja activa", _
                Default:=HojaTest.Name)
   '  Sale del procedimiento si el usuario hace clic en el
   '  botón Cancelar o no indica ningún nombre
      If NuevoNombre = "" Then Exit Sub
      HojaTest.Name = NuevoNombre
   Next
   '  Desactiva la administración de errores
   On Error GoTo 0
   '  Selecciona la primera hoja y guarda el libro
   Sheets(1).Select
   ActiveWorkbook.Save
   Exit Sub

   '  Rutina de administración de errores que muestra
   '  un mensaje y reanuda en la línea número 1
   AdministracionDeErrores:
      MsgBox "Nombre de hoja incorrecto o existente"
      Resume 1
End Sub
```

© Editions ENI - All rights reserved

Copyrighted material

Hidden page

HelpFile

Devuelve una cadena de caracteres que contiene la ruta de acceso completa del archivo de ayuda.

LastDLLError

Devuelve un código de error de sistema producido por una llamada a una biblioteca de vínculos dinámicos (DLL).

Number

Devuelve o establece un valor numérico que especifica el número del error.

Source

Devuelve o establece una cadena de caracteres que especifica el nombre del objeto o la aplicación que generó el error.

Métodos

Clear

Borra todas las propiedades establecidas del objeto **Err**.

Raise

Permite generar errores de ejecución.

Ejemplo

El siguiente código muestra un mensaje que brinda información sobre la naturaleza del error.

```
Private Sub Workbook_SheetActivate(ByVal Sh As Object)
   Dim rng As Range

   '  Muestra un mensaje si la hoja activada
   '  es el resumen de ventas
   On Error GoTo Error

   If Sh.Name = "Resumen" Then
      MsgBox "Resumen de las diferentes hojas ", vbExclamation
   Set rng = Worksheets("Enero").Range("A1:G11")
   End If                                                    .../...
```

© Editions ENI - All rights reserved

Copyrighted material

Hidden page

Hidden page

D. Comunicación con Outlook desde Excel 288

1. El modelo de objetos Outlook 288
2. Ejemplo 289

E. El protocolo DDE 290

1. La inicialización 291
2. La vinculación 291
3. La clausura 293

F. Objetos vinculados o incrustados 294

1. Métodos del objeto OLEObject 295
2. Propiedades del objeto OLEObject 295

G. Métodos y propiedades relativos a los vínculos con Excel 296

1. Métodos y propiedades del objeto Workbook 296
2. Métodos y propiedades de otros objetos 299

Copyrighted Material

A. La tecnología Automation

1. Presentación

Automation, llamada también **OLE** (*Object Linking and Embedding*) u OLE Automation, es una tecnología que permite manipular los objetos de otra aplicación directamente desde Excel o de VBA Excel.

Para poder funcionar, **Automation** necesita un cliente y un servidor, llamado **servidor OLE**. El servidor es la aplicación o el componente que realiza los servicios al cliente. El cliente (también llamado controlador) usa estos servicios para pilotar la aplicación servidor y manipular sus objetos. Por ejemplo, si se ejecuta una combinación de correspondencia Word desde VBA Excel, Excel es el cliente y Word el servidor OLE.

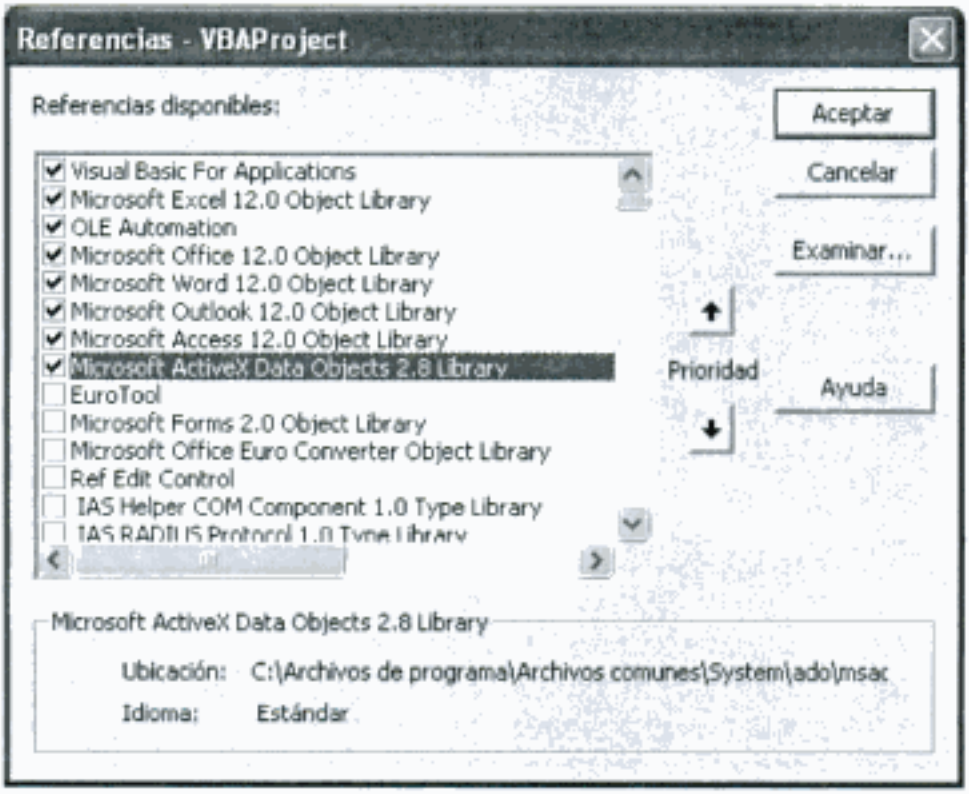
Una **biblioteca de objetos** es un archivo, generalmente con extensión olb, que provee la información para manipular los objetos puestos a disposición por un servidor. Se puede usar el Examinador de objetos para examinar el contenido de una biblioteca de objetos.

Para tener acceso a los objetos de otra aplicación, se debe referenciar su biblioteca de objetos de la siguiente manera:

- Seleccione la opción **Referencias** del menú **Herramientas**. Aparece el cuadro de diálogo **Referencias - VBAProject** con todos los servidores OLE guardados en la base del registro.

© Editions ENI - All rights reserved

Copyrighted material



→ Active a continuación las referencias deseadas.

El número indicado con el nombre de la biblioteca de objetos corresponde a la versión de Microsoft Office (9.0 para la versión 2000, 10.0 para la versión 2002, 11.0 para la versión 2003 y 12.0 para la versión 2007).

2. Uso de la tecnología Automation

Para manipular los objetos de otra aplicación, proceda de la siguiente manera:

→ Defina en el código VBA una variable objeto.

→ Use las funciones **CreateObject** o **GetObject** para hacer referencia al objeto.

Ejemplos

Ejecutar Word

```
Dim appWord as Object
Set appWord = CreateObject("Word.Application")
```

o

```
Dim appWord As New Word.Application
```

Referencia a un documento Word existente

```
Dim docWord As New Word.Document
Set docWord = GetObject("C:\Clientes\mailing.doc")
```

Si hay instalada más de una versión del mismo programa, se ejecutará la última versión guardada en la base de registro.

Sin embargo, usted puede precisar la versión del programa que desea ejecutar:

Ejemplo

Iniciar la aplicación Word 2000

```
Dim appWord as Object
Set appWord = CreateObject("Word.Application.10")
```

Los siguientes párrafos describen cómo manejar distintos programas de Microsoft Office usando la tecnología Automation.

Los objetos, colecciones, métodos y propiedades de los modelos de objeto de Microsoft Office son muy numerosos. Solamente se describen a continuación los más usados.

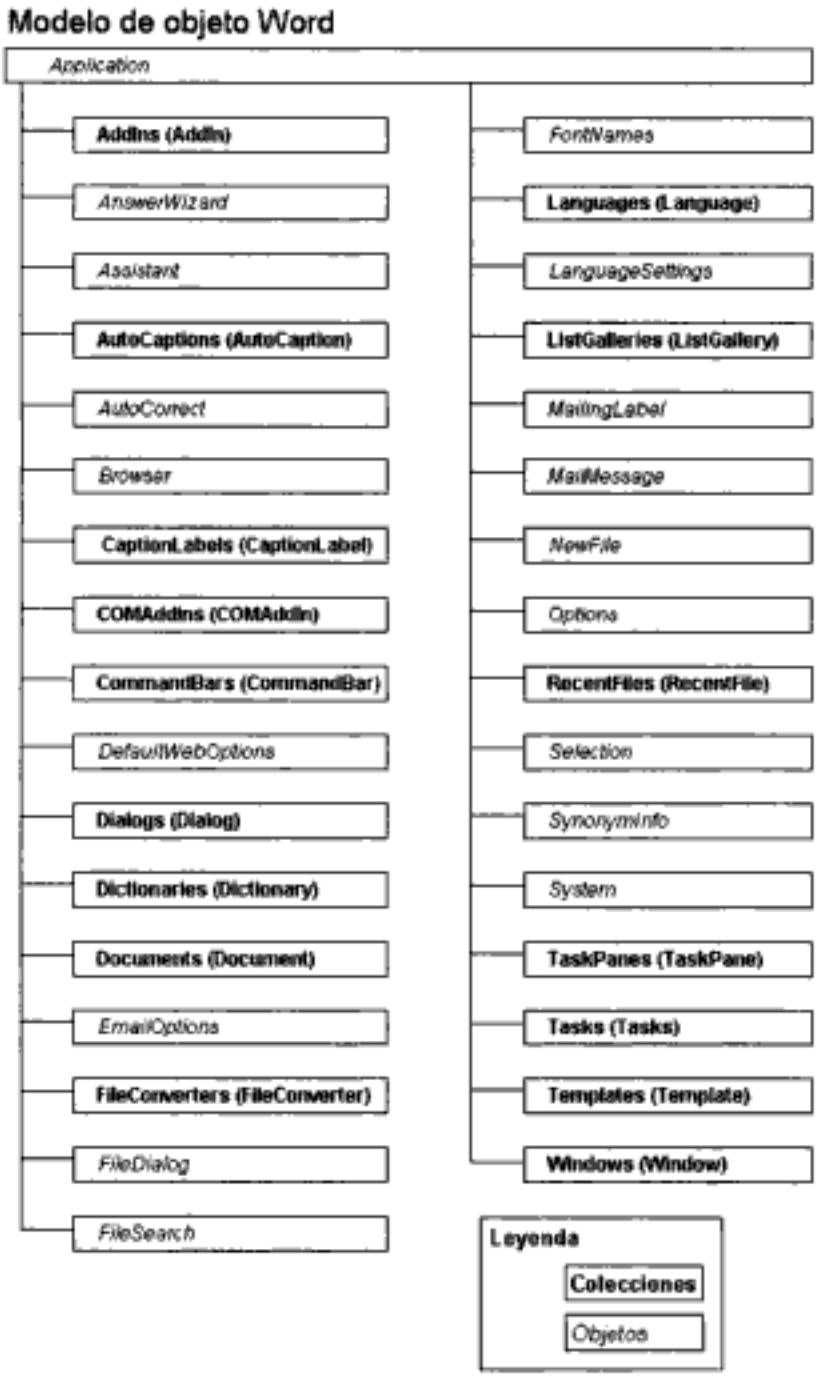
© Editions ENI - All rights reserved

Copyrighted material

B. Comunicación con Word desde Excel

1. El modelo de objeto Word

Extracto del modelo de objeto Word:



Copyrighted Material

2. Objetos y colecciones Word

Documents

Colección de todos los archivos Word (objetos **Document**) abiertos.

Dictionaries

Colección de los diccionarios personales activos (objetos **Dictionary**).

EmailOptions

Objeto que contiene los atributos globales que Microsoft Word usa cuando se crean y modifican mensajes de correo electrónico y las respuestas a esos mensajes.

FileConverters

Colección de los convertidores de archivo (objetos **FileConvert**) disponibles para abrir y guardar archivos.

FontNames

Objeto que contiene la lista de nombres de las fuentes disponibles.

Languages

Colección de los idiomas (objetos **Language**) usados en Word para la comprobación lingüística y el formato.

ListGalleries

Colección de las bibliotecas de modelos de lista (con viñetas, números, jerarquización).

MailMessage

Objeto que representa el mensaje de correo electrónico activo si Word es su editor de correo electrónico.

Options

Representa las opciones de aplicación y de documento de Word.

Selection

Representa la selección actual en una ventana o un panel.

© Editions ENI - All rights reserved

Copyrighted Material

System

Contiene información sobre el sistema de su ordenador.

Templates

Colección de las plantillas (objetos **Template**) actualmente disponibles.

3. La colección Documents

La colección **Documents** está formada por los objetos **Document** abiertos en Word.

Métodos

Add

Crea un nuevo documento y lo agrega a la colección **Documents**.
Ejemplo: Documents.Add Template:="Normal"

Close

Cierra todos los documentos Word abiertos.
Ejemplo: Documents.Close

Open

Abre el documento especificado y lo agrega a la colección **Documents**.
Ejemplo: Documents.Open FileName:="C:\Clientes\Informe.doc", ReadOnly:=True

Save

Guarda todos los documentos abiertos.
Ejemplo: Documents.Save

4. El objeto Document

El objeto **Document** hace referencia a un documento Word. **ActiveDocument** designa al documento activo.

Copyrighted Material

Métodos

Activate

Activa un documento ya abierto.
Ejemplo: Documents("Compras.doc").Activate

Close

Cierra un documento Word abierto.
Ejemplo: Documents("Compras.doc").Close
O ActiveDocument.Close

PrintPreview

Muestra la vista preliminar de un documento.
Ejemplo: ActiveDocument.PrintPreview

Range

Devuelve un objeto **Range** (Selection).
Ejemplo: ActiveDocument.Range(0, 50).Bold = True

Save

Guarda un documento.
Ejemplo: ActiveDocument.Save

SaveAs

Guarda un documento con un nuevo nombre o en otro formato.
Ejemplo: ActiveDocument.SaveAs FileName:=strDocName

Objetos y colecciones

Characters

Colección de los caracteres de un documento.

MailMerge

Fusión y combinación de correspondencia en Word.

PageSetup

Opciones de configuración de página.

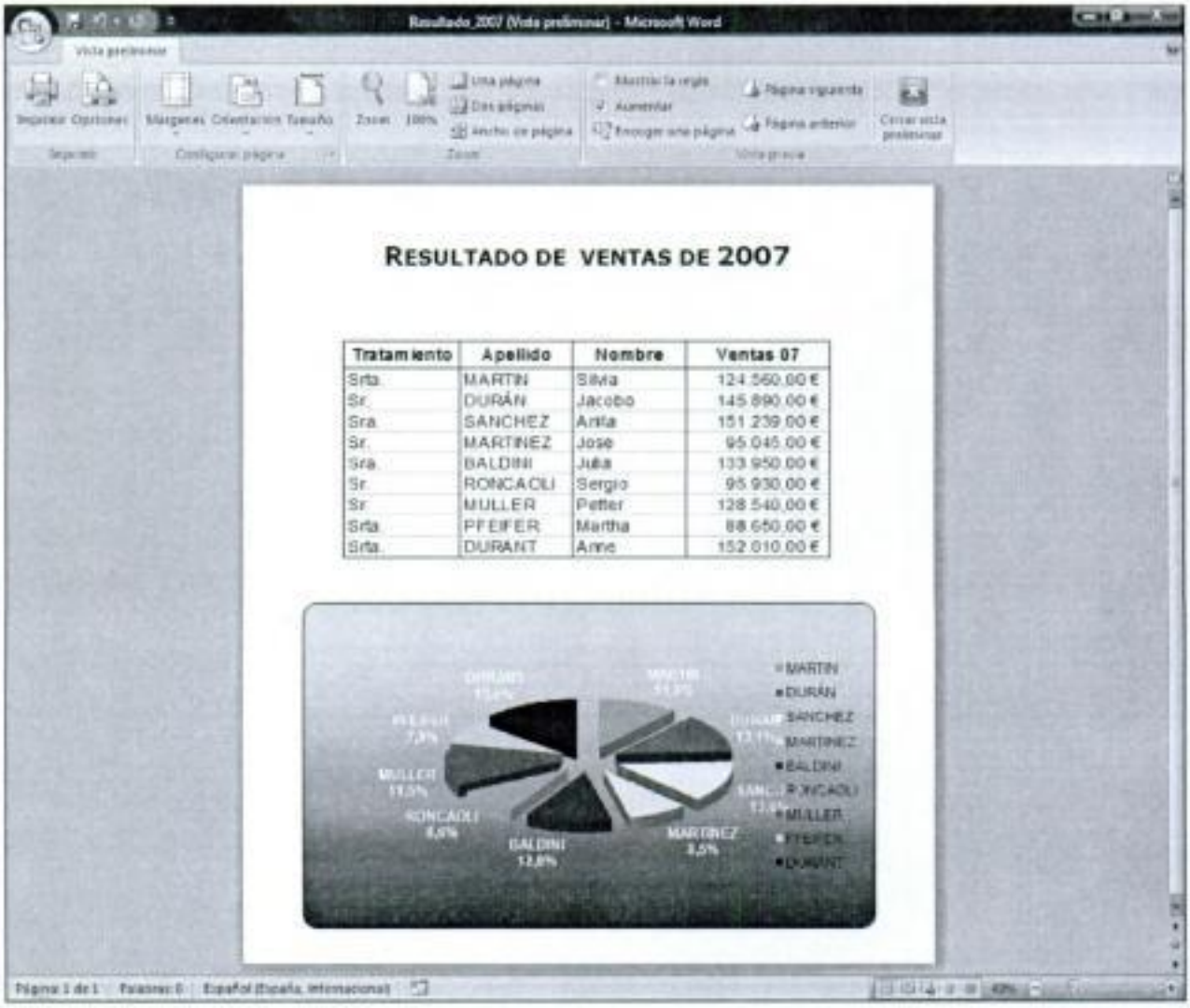
© Editions ENI - All rights reserved

Copyrighted material

Hidden page

Hidden page

Resultado en Word:



Tratamiento	Apellido	Nombre	Ventas 07
Srta.	MARTIN	Silvia	124.560,00 €
Sr.	DURÁN	Jacobo	145.890,00 €
Sra.	SANCHEZ	Anita	151.239,00 €
Sr.	MARTINEZ	Jose	95.045,00 €
Sra.	BALDINI	Julia	133.950,00 €
Sr.	RONCADOLI	Sergio	95.930,00 €
Sr.	MULLER	Peter	128.540,00 €
Srta.	PFEIFER	Martha	88.650,00 €
Srta.	DURANT	Anne	152.010,00 €

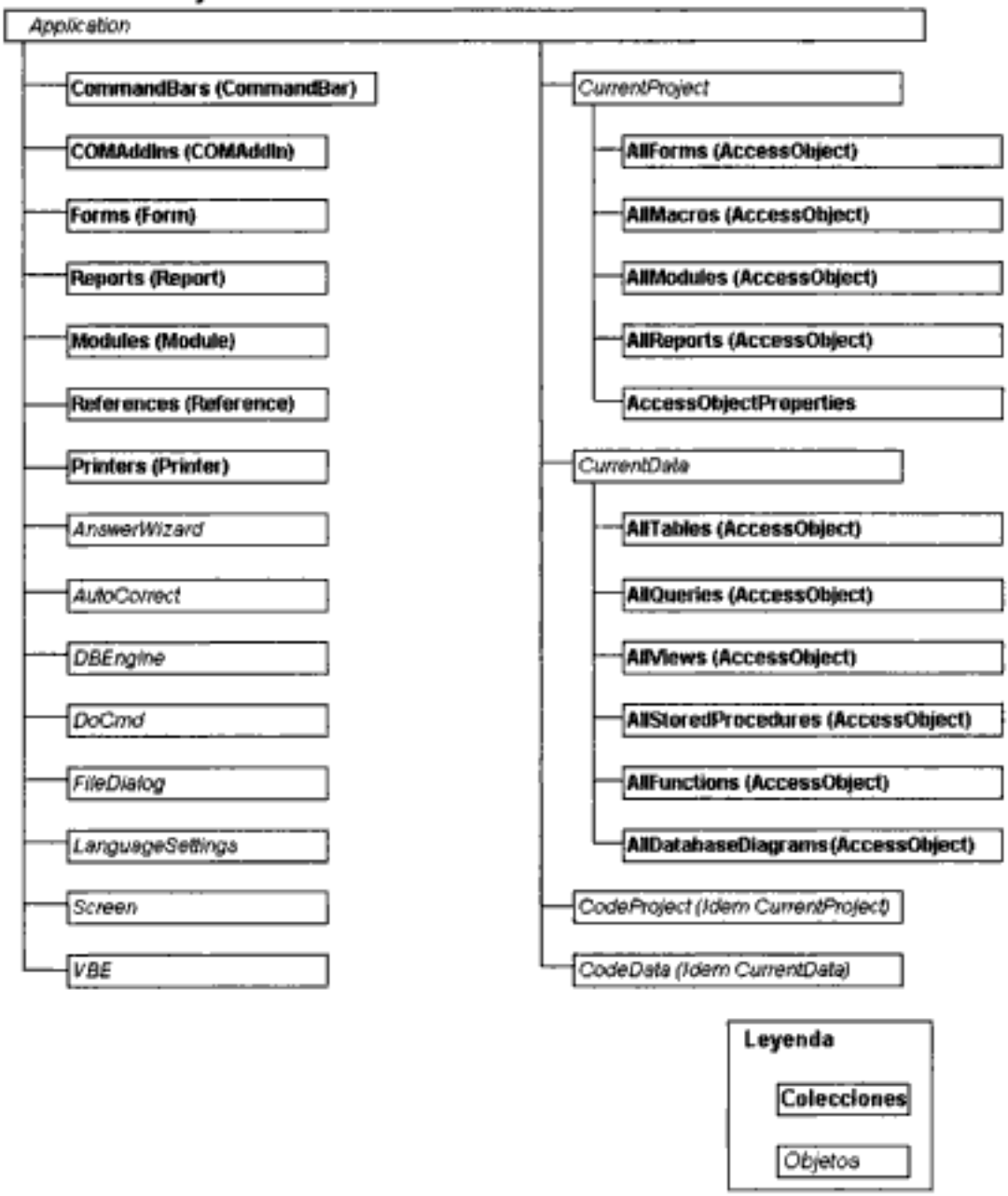
Copyrighted material

C. Comunicación con Access desde Excel

1. El modelo de objeto Access

Extracto del modelo de objeto Access:

Modelo de objeto de Microsoft Access



© Editions ENI - All rights reserved

Copyrighted material

2. Ejemplos

a. Listar tablas de una base Access

Este ejemplo muestra cómo ver la lista de tablas de la base Access Northwind 2007.accdb.

```
Sub Tablas_Access()
Dim appAccess As Access.Application
Dim i, j As Integer

'   Inicia una sesión Microsoft Access
Set appAccess = CreateObject("Access.application")

'   Muestra la lista de tablas recorriendo
'   la colección Alltables del objeto CurrentData
With appAccess
   .OpenCurrentDatabase (ThisWorkbook.Path & "\Northwind 2007.accdb")
   j = 6
   For i = 1 To .CurrentData.AllTables.Count - 1
       If Left(UCase(.CurrentData.AllTables(i).Name), 4) _
          <> "MSYS" Then
          Range("A" & j) = .CurrentData.AllTables(i).Name
          j = j + 1
       End If
   Next i
End With

'   Sale de la aplicación Access
appAccess.Quit

'   Reinicializa el objeto
Set appAccess = Nothing
End Sub
```


Copyrighted Material

b. Mostrar una tabla Access en Excel

El objeto QueryTable representa un rango de datos externo (base de datos, página Web, etc.) contenida en una hoja de cálculo. Las propiedades de este objeto se explican en el capítulo 11, dedicado a Internet.

```
Sub Mostrar_Tabla()
Dim i As Integer

'   Borra los datos de la última tabla mostrada
If ActiveSheet.ListObjects.Count > 0 Then
  For i = ActiveSheet.ListObjects(1).ListColumns.Count To 1 Step -1
    ActiveSheet.ListObjects(1).ListColumns(i).Delete
  Next i
End If

'   Muestra el contenido de la tabla seleccionada
'   haciendo una consulta en la base Northwind 2007
On Error GoTo 1:
If ActiveCell = "" And ActiveCell.Column = 3 Then
   With ActiveSheet.ListObjects.Add(SourceType:=0, Source:=Array( _
        "OLEDB;Provider=Microsoft.ACE.OLEDB.12.0;", _
        "Data Source=" & ThisWorkbook.Path & "\Northwind 2007.accdb;", _
        "Mode=Share Deny Write;Jet OLEDB:Engine Type=6"), _
        Destination:=Range("$D$2")).QueryTable
        .CommandType = xlCmdTable
        .CommandText = Array(ActiveCell)
        .RowNumbers = False
        .PreserveFormatting = True
        .BackgroundQuery = True
        .RefreshStyle = xlInsertDeleteCells
        .AdjustColumnWidth = True
        .PreserveColumnInfo = True
        .AdjustColumnWidth = True
        .PreserveColumnInfo = True
        .Refresh BackgroundQuery:=False
    End With
Else
    MsgBox "Debe seleccionar un nombre de tabla", vbExclamation
End If
On Error GoTo 0
Exit Sub                                                    .../...
```

© Editions ENI - All rights reserved

Copyrighted material

Hidden page

c. Abrir una tabla o consulta Access en un nuevo libro

El método OpenDatabase usado permite importar una tabla o el resultado de una consulta Access en un nuevo libro Excel.

```
Sub Abrir_Tabla()
   Dim TableName As String
   Dim wbk As Workbook
   Dim wbkName As String
   Dim fso As Object
   '  Controla la celda seleccionada
   If ActiveCell = "" Or ActiveCell.Column <> 3 Then Exit Sub
   '  Abre la tabla seleccionada
   TableName = ActiveCell
   Set wbk = Workbooks.OpenDatabase( _
       Filename:=ThisWorkbook.Path & "\Northwind 2007.accdb", _
       CommandText:=Array(TableName), CommandType:=xlCmdTable)
   '  Controla si el archivo Excel ya existe
   wbkName = ThisWorkbook.Path & "\" & TableName & ".xlsx"
   If Dir(wbkName) <> "" Then
      If MsgBox("Libro " & TableName _
         & " ya existente, ¿desea eliminarlo?", _
         vbQuestion + vbYesNo) = vbYes Then
         Set fso = CreateObject("Scripting.FileSystemObject")
         fso.deletefile wbkName
         Set fso = Nothing
      Else
         Exit Sub
      End If
   End If   '  Guarda y cierra el libro creado
   wbk.SaveAs wbkName
   wbk.Close
End Sub
```

Los datos importados también pueden actualizarse en todo momento.

© Editions ENI - All rights reserved

Copyrighted material

Hidden page

2. Ejemplo

Este ejemplo permite crear mensajes (mails) Outlook y enviar el libro activo como dato adjunto.

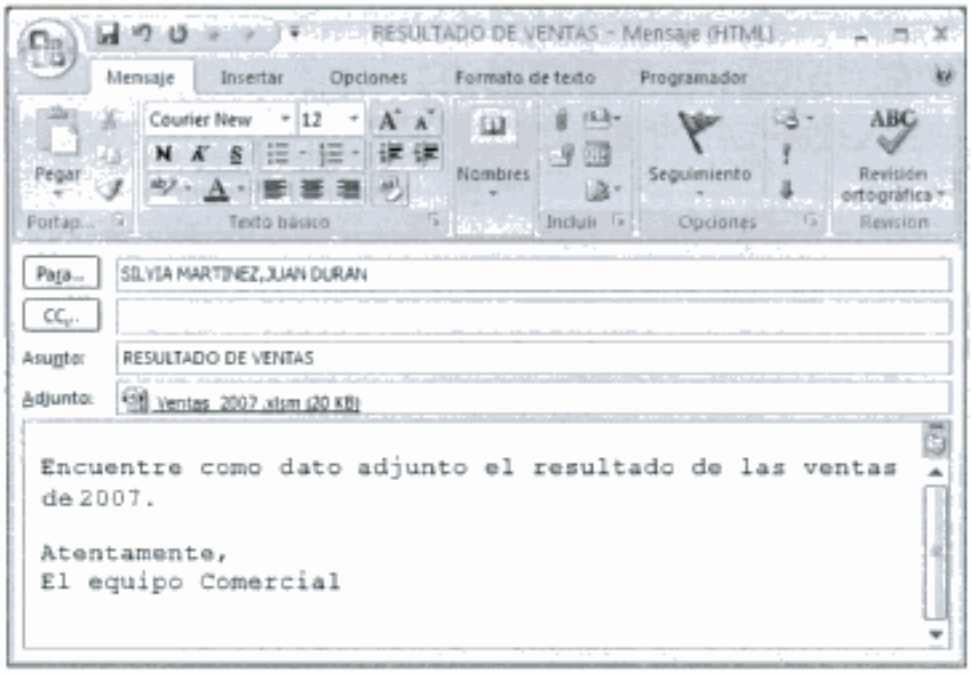
```
Sub Enviar_Email()
Dim appOutlook As Outlook.Application
Dim message As Outlook.MailItem
Dim myRecipient As Object

' Abre una sesión Microsoft Outlook
Set appOutlook = CreateObject("outlook.application")
' Crea un nuevo mensaje
Set message = appOutlook.CreateItem(olMailItem)

With message
 '  Asunto, Texto, Destinatarios, datos adjuntos del mensaje
 .Subject = "RESULTADO DE VENTAS"
 .Body = "Encuentre como dato adjunto el resultado de las ventas de 2007." _
         & Chr(13) & "Atentamente," & Chr(13)  _
         & "El equipo Comercial"
 .BodyFormat = olFormatHTML
 .Recipients.Add ("SILVIA MARTINEZ")
 .Recipients.Add ("JUAN DURAN")
 .Attachments.Add ThisWorkbook.Path & "\" & ThisWorkbook.Name

 '  Envía el mensaje
 .Send
End With

' Sale de la aplicación Outlook
appOutlook.Quit

' Reinicializa el objeto
Set appOutlook = Nothing
End Sub
```

Para probar este ejemplo, reemplace los nombres Silvia Martínez y Juan Durán por nombres que figuren en su lista de contactos de Outlook.

© Editions ENI - All rights reserved

Copyrighted Material

Resultado en Outlook:



E. El protocolo DDE

DDE (*Dynamic Data Exchange*) es un protocolo de intercambio dinámico de datos entre dos aplicaciones Windows, una llamada **Cliente** o **Destination**, la otra llamada **Servidor** o **Source**. El vínculo se rompe si una de las dos aplicaciones se cierra.

Un vínculo DDE comprende tres fases principales:

- **inicialización**: la aplicación Destination busca la aplicación Source y establece con ella un canal de comunicación (vínculo).
- **conversación**: se intercambian los datos en el canal.
- **clausura**: se cierra el canal de comunicación.

Copyrighted Material

Hidden page

➔ DDESend

La función **DDESend** entabla una conversación **DDE** con otra aplicación, envía un elemento de información a esta aplicación desde un control en un formulario o en un informe.

Sintaxis

```
DDESend(<Aplicación>, <Rúbrica>, <Elemento>, <Datos>)
```

Aplicación	Expresión de cadena que identifica una aplicación.
Rúbrica	Archivo documento o archivo de datos.
Elemento	Referencia a un dato de la aplicación Source.
Datos	Cadena o expresión que contiene los datos que hay que enviar a la aplicación Destination.

➔ DDEPoke

La instrucción **DDEPoke** envía información a la aplicación Source por un canal abierto.

Sintaxis

```
DDEPoke(<Número de canal>, <Elemento>, <Dato>)
```

Número de canal	N° de canal devuelto por la función **DDEInitiate**.
Elemento	Referencia a un dato de la aplicación Source.
Dato	Dato que hay que pasar a la aplicación Source.

Copyrighted material

➔ DDERequest

La función **DDERequest** pide información a la aplicación Source.

Sintaxis

```
DDERequest(<Número de canal>, <Elemento>)
```

Número de canal	N° de canal devuelto por la función **DDEInitiate**.
Elemento	Referencia a un dato de la aplicación Source.

➔ DDEExecute

La instrucción **DDEExecute** envía una cadena de comandos a la otra aplicación que se encuentra vinculada.

Sintaxis

```
DDEExecute(<Número de canal>, <Comando>)
```

Número de canal	N° de canal devuelto por la función **DDEInitiate**.
Comando	Cadena que contiene un comando reconocido por la otra aplicación.

3. La clausura

➔ DDETerminate

La instrucción **DDETerminate** cierra el canal de comunicación DDE.

Sintaxis

```
DDETerminate(<Número de canal>)
```

© Editions ENI - All rights reserved

Copyrighted material

Número de canal	N° de canal devuelto por la función **DDEInitiate**.

➔ DDETerminateAll

La instrucción **DDETerminateAll** cierra todos los canales de comunicación DDE.

Sintaxis

```
DDETerminateAll
```

Ejemplo

Este ejemplo muestra cómo copiar celdas Excel al principio de un documento Word.

```
Sub TransferirWord()
Dim Canal As Variant
'  Abre un canal hacia Word y abre el documento Empleados.doc
Canal = DDEInitiate(app:="WinWord", _
       topic:=ThisWorkbook.Path & "\Empleados.docx")
'  Selecciona un rango con nombre
Worksheets("Transferir DDE").Range("Empleados").Select
'  Copia el contenido de la selección al abrir el documento Word
DDEPoke Canal, "\EndOfDoc", Selection
'  Termina el intercambio DDE
DDETerminate Canal
End Sub
```

F. Objetos vinculados o incrustados

Es posible manipular los objetos vinculados o incrustados en Excel a través de la colección **OLEObjects** de objetos **OLEObject**.

El objeto emparentado puede ser un objeto **Worksheet** o un objeto **Chart**.

Copyrighted material

Hidden page

Application	Name
AutoUpDate	Object
Border	Parent
BottomRightCell	Placement
Count	PrintObject
Creator	ProgId
Enabled	ShapeRange
Height	Shadow
Index	SourceName
Interior	Top
Left	TopLeftCell
LinkedCell	Visible
ListFillRange	Width
Locked	ZOrder

G. Métodos y propiedades relativos a los vínculos con Excel

1. Métodos y propiedades del objeto Workbook

➔ ChangeLink

(método)

Modifica un vínculo entre dos documentos.

```
ChangeLink (NAME,NEWNAME,type)
```

NAME	Nombre del vínculo que se debe modificar.
NEWNAME	Nuevo nombre del vínculo.
Type	Tipo de vínculo que hay que devolver: xlLinkTypeExcelLinks, xlLinkTypeOLELinks.

Copyrighted Material

➔ LinkInfo

(método)

Devuelve información acerca de la fecha y el estado de actualización del vínculo.

`LinkInfo(NAME,LINKINFO,Type,EditionRef)`

NAME	Nombre del vínculo.
LINKINFO	Tipo de información que hay que devolver: (xlUpdateState o xlEditionDate).
Type	Tipo de vínculo que se ha de devolver: (xlLinkInfoOLELinks, xlLinkInfoPublishers o xllinkInfoSubscribers).
EditionRef	Si el vínculo es una edición, este argumento especifica la referencia a la edición.

➔ LinkSources

(método)

Devuelve una matriz Visual Basic con los vínculos del libro.

`LinkSources(type)`

Type	Tipo de vínculo que hay que devolver: xlExcelLinks, xlOLELinks, xlPublishers, xlSubscribers

➔ OpenLinks

(método)

Abre el documento Source de un vínculo.

`OpenLinks(NAME,readonly,type)`

© Editions ENI - All rights reserved

Copyrighted material

NAME	Nombre del vínculo.
Readonly	Apertura en modo sólo lectura (**True** o **False**).
Type	Tipo del vínculo: (xlExcelLinks, xlOLELinks, xlPublishers, o xlSubscribers).

➔ SaveLinkValues

(propiedad)

Propiedad que devuelve **True** si Microsoft Excel guarda los valores de los vínculos externos con el libro.

➔ SetLinkOnData

(método)

Crea un procedimiento que se ejecutará cada vez que se actualice un vínculo DDE.

```
SetLinkOnData(NAME,PROCEDIMIENTO)
```

NAME	Nombre del vínculo.
PROCEDIMIENTO	Nombre del procedimiento que se debe ejecutar cuando se actualiza el vínculo.

➔ UpDateLink

(método)

Actualiza un vínculo.

```
UpDateLink(NAME,type)
```

NAME	Nombre del vínculo.

Copyrighted material

`Type` Tipo de vínculo (`xlLinkTypeExcelLinks` o `xlLinkTypeOLELinks`).

2. Métodos y propiedades de otros objetos

➔ AskToUpDateLinks

(propiedad del objeto **Application**)

Propiedad que devuelve **True** si Excel invita al usuario a actualizar los vínculos cuando se abren los archivos que los contienen.

➔ LinkSource

(propiedad del objeto **DocumentProperty**)

Devuelve o define el origen de una propiedad de un documento personalizado con vínculos.

➔ LinkToContent

(propiedad del objeto **DocumentProperty**)

Propiedad que devuelve **True** si el valor de una propiedad de documento personalizada está vinculada al contenido del documento contenedor.

➔ ActivateMicrosoftApp

(método del objeto **Application**)

Activa una aplicación Microsoft.

Application.ActivateMicrosoftApp(INDEX)

`INDEX` Aplicación Microsoft que hay que activar. Algunos ejemplos: `xlMicrosoftWord`, `xlMicrosoftAccess`, `xlMicrosoft Project`, etc.

© Editions ENI - All rights reserved

Copyrighted material

Copyrighted material

Capítulo 11: Internet

A. Consultas por Internet 303

B. El objeto QueryTable 305

1. Propiedades del objeto QueryTable 306

2. Ejemplos 311

C. Publicación de páginas Web 312

1. Asociación de un elemento de libro a una página Web . . . 313

2. Publicación de la página Web 314

3. Ejemplo 314

D. Los objetos WebOptions y DefaultWebOptions 315

1. Propiedades 316

a. Opciones de la ficha General 316

b. Opciones de la ficha Exploradores 317

c. Opciones de la ficha Archivos 318

d. Otras propiedades 318

2. Método del objeto WebOptions 319

Copyrighted material

E. Importación, exportación y asignación de archivos XML 319

1. Colecciones 320
2. Métodos del objeto Workbook 321
3. Eventos del objeto Workbook 321
4. Métodos del objeto XmlMap 322

F. El objeto HyperLink 323

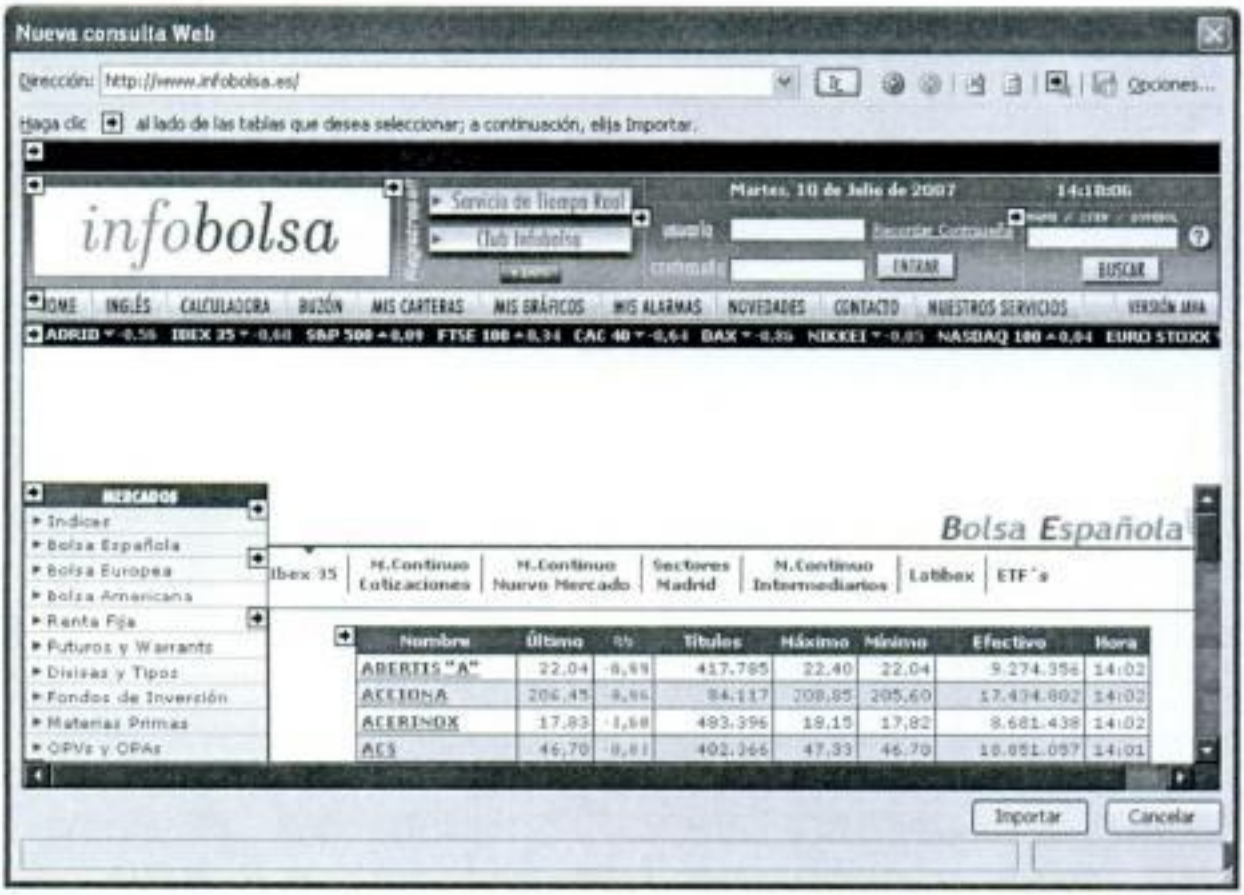
1. Propiedades 323
2. Métodos 324

Copyrighted Material

Hidden page

Ejemplo:

El siguiente ejemplo importa las cotizaciones de la Bolsa de Valores de Madrid desde el sitio www.infobolsa.es.



Copyrighted material

Resultado en Excel:

Capitulo11.xlsm

	A	B	C	D	E	F	G
1	Nombre	Último	%	Títulos	Fecha	Hora	
2	ADIDAS AG	46,95	-0,8	599.862	10-jul	13:58	
3	ALLIANZ VN/	171,72	0,09	2.964.504	10-jul	13:58	
4	BASF AG	97,68	-1,49	1.328.226	10-jul	13:58	
5	BAYER AG	55,04	-1,57	2.824.736	10-jul	13:58	
6	BMW STAMI	48,46	-1,9	1.528.023	10-jul	13:57	
7	CO.BK AG	34,94	-2,02	2.499.825	10-jul	13:58	
8	CONTINENTA	106,15	-1,26	469.597	10-jul	13:57	
9	DAIMLERCHI	69,14	0,3	8.092.146	10-jul	13:58	
10	DEUTSCHE E	63,26	-1,51	329.477	10-jul	13:57	
11	DT.BANK NA	106,49	-1,88	2.318.704	10-jul	13:59	
12	DT.BOERSE	87,4	0,49	931.964	10-jul	13:59	
13	DT.POST NA	23,67	0,17	3.612.221	10-jul	13:58	
14	DT.TELEKOM	13,63	-0,73	7.991.752	10-jul	13:59	
15	E.ON AG	121,25	-0,91	1.866.868	10-jul	13:59	
16	FRESEN.ME	33,68	-0,77	960.526	10-jul	13:58	
17	HENKEL VZ	41,19	-0,24	729.766	10-jul	13:57	
18	HYPO REAL	46,55	1,92	639.249	10-jul	13:57	

Acciones

B. El objeto QueryTable

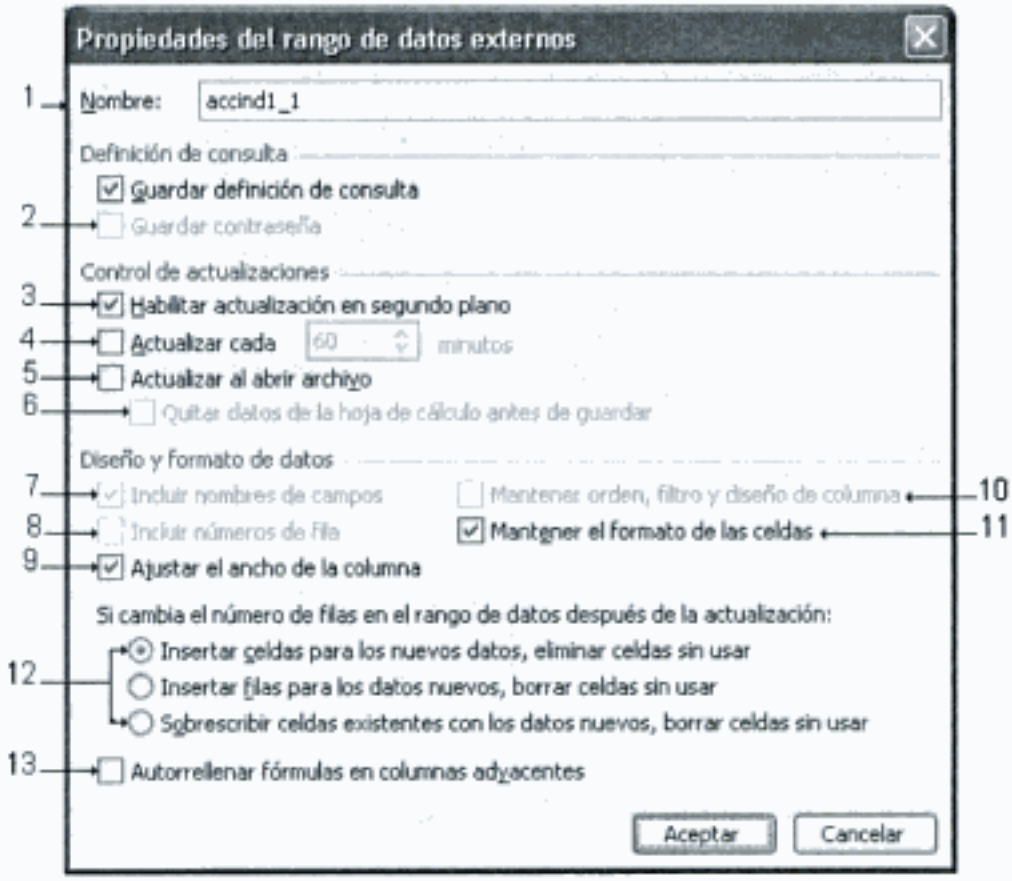
El objeto **QueryTable** (tabla de consulta) representa un rango de datos externos contenido en una hoja de cálculo. Estos datos pueden provenir de un origen externo, como una base de datos Microsoft Access o SQL Server o de datos extraídos con una **consulta Web**.

El objeto **QueryTable** pertenece a la colección **QueryTables** del objeto **Worksheet**.

© Editions ENI - All rights reserved

Copyrighted material

Hidden page



N°	Propiedades	Valores devueltos
1.	Name	Cadena de caracteres
2.	SavePassword	Boolean
3.	BackgroundQuery	Boolean
4.	RefreshPeriod	Entero largo
5.	RefreshOnFileOpen	Boolean
6.	SaveData	Boolean
7.	FileNames	Boolean
8.	RowNumbers	Boolean
9.	AdjustColumnWidth	Boolean
10.	PreserveColumnInfo	Boolean
11.	PreserveFormatting	Boolean
12.	RefreshStyle	Constante
	Constantes	**xlInsertDeleteCells** **xlInsertEntireRows** **xlOverwriteCells**
13.	FillAdjacentFormulas	Boolean

© Editions ENI - All rights reserved

Hidden page

5. WebDisableDateRecognition Boolean
6. WebDisableRedirections Boolean

Otras propiedades usadas por las consultas Web

Connection

Cadena de caracteres. URL del origen de datos Web.

Destination

Objeto **Range**. Devuelve la celda de la esquina superior izquierda del rango de datos externo.

EditWebPage

Variant. URL de la página Web.

EnabledEditing

Boolean. Indica si el usuario puede modificar la consulta.

EnableRefresh

Boolean. Indica si el usuario puede actualizar los datos del rango de datos externo.

MaintainConnection

Boolean. Indica si la conexión al origen de datos externos se mantiene después de la actualización y hasta que se cierra el libro.

Name

Nombre del rango de origen de datos externos.

QueryType

Constante. Indica el tipo de consulta que Microsoft Excel utiliza para rellenar el origen de datos externos (**xlWebQuery** para las consultas Web).

ResultRange

Objeto **Range**. Representa el área de la hoja de cálculo ocupada por el origen de datos externos.

© Editions ENI - All rights reserved

Copyrighted Material

WebSelectionType

Constante. Establece un valor que determina qué parte de la página Web se importará: toda la página, todas las tablas de la página o sólo algunas de ellas.

Constantes **xlWebEntirePage**
xlWebAllTables
xlWebSpecifiedTables

WebTables

Cadena de caracteres. Lista, delimitada por comas, de los nombres o números de índice de las tablas que hay que importar.

Métodos

CancelRefresh

Cancela todas las consultas en segundo plano del rango de datos externo especificado.

Delete

Elimina el rango de datos externo especificado.

Refresh

Actualiza el rango de datos externo.

ResetTimer

Restablece el temporizador automático de actualización utilizando el último intervalo establecido mediante la propiedad **RefreshPeriod** (frecuencia de actualización).

SaveAsODC

Guarda el origen de los datos externos como archivo de conexión de datos de Microsoft Office. (extensión .odc). No funciona con las consultas Web.

Copyrighted Material

Hidden page

```
.../...
        .WebDisableDateRecognition = False
        .WebDisableRedirections = False
            '   Actualiza los datos
        .Refresh BackgroundQuery:=False
    End With
End Sub
```

Ejemplo de llamado del procedimiento : importar la tabla número 13 de la página web www.infobolsa.es/v2002/home/home.asp en la hoja "Acciones" del libro activo.

```
Sub ImportarPalmares()
    Dim rng As Range
    Dim Url As String
    Dim Name As String
    Dim i As Integer
    '   Cancela las consultas existantes y borra los datos
    If ActiveSheet.QueryTables.Count > 0 Then
       For i = ActiveSheet.QueryTables.Count To 1 Step -1
           ActiveSheet.QueryTables(i).Delete
           Range("A1:X100").Delete
       Next i
    End If
    '   Llama al procedimiento CreateQueryWeb
    Url =  "URL;http://www.infobolsa.es/v2002/home/home.asp"
    Name = "Home"
    Set rng = ActiveWorkbook.Worksheets("Acciones").Range("A1")
    CreateWebQuery Url, Name, rng, 13
End Sub
```

C. Publicación de páginas Web

Es posible crear y publicar una página Web a partir de un libro, de una hoja Excel, de un gráfico, de un rango de celdas...

Para publicar una página Web a partir de Excel, haga clic en el botón Office y luego en **Guardar como** - **Otros formatos**. Dentro de la opción **Tipo**, haga clic en **Página Web**. Haga clic en el botón **Publicar** para modificar los parámetros de publicación.

Copyrighted material

En VBA, para asociar un elemento de un libro a una página Web, se debe crear un objeto **PublishObject** (usando el método **Add** de la colección **Publish Objects**). Para publicar la página Web, luego deberá usar el método **Publish** del objeto **PublishObject**.

1. Asociación de un elemento de libro a una página Web

Sintaxis

```
PublishObjects.Add(SourceType, FileName, Sheet, Source,
HtmlType, DivID, Title)
```

Solamente los argumentos `SourceType` y `FileName` son obligatorios.

`PublishObjects`	Expresión que devuelve una colección **PublishObjects**.
`SourceType`	Tipo de elemento que se debe publicar (xlSourceSheet, SourceRange, xlSourceworkbook, xlSourceChart, xlSource-Query, xlSourcePivotTable...).
`Sheet`	Nombre de la hoja de cálculo guardada como página Web.
`Source`	Nombre del elemento que hay que publicar si se trata de un gráfico, de un informe de tabla dinámica o de una tabla de consulta.
`HtmlType`	Especifica si el elemento publicado se guarda como componente de Microsoft Office Web interactivo o como texto e imágenes estáticas.
`DivId`	Identificador exclusivo utilizado en la etiqueta DIV de HTML para identificar el elemento en la página Web.
`Title`	Título de la página Web.

© Editions ENI - All rights reserved

Copyrighted material

2. Publicación de la página Web

Sintaxis

```
PublishObject.Publish(Create)
```

`PublishObject`	Expresión que devuelve un objeto **PublishObject** o una colección **PublishObjects**.
`Create`	Si este argumento tiene el valor **True** y el archivo HTML ya existe, el mismo es reemplazado. El valor por defecto es **False**.

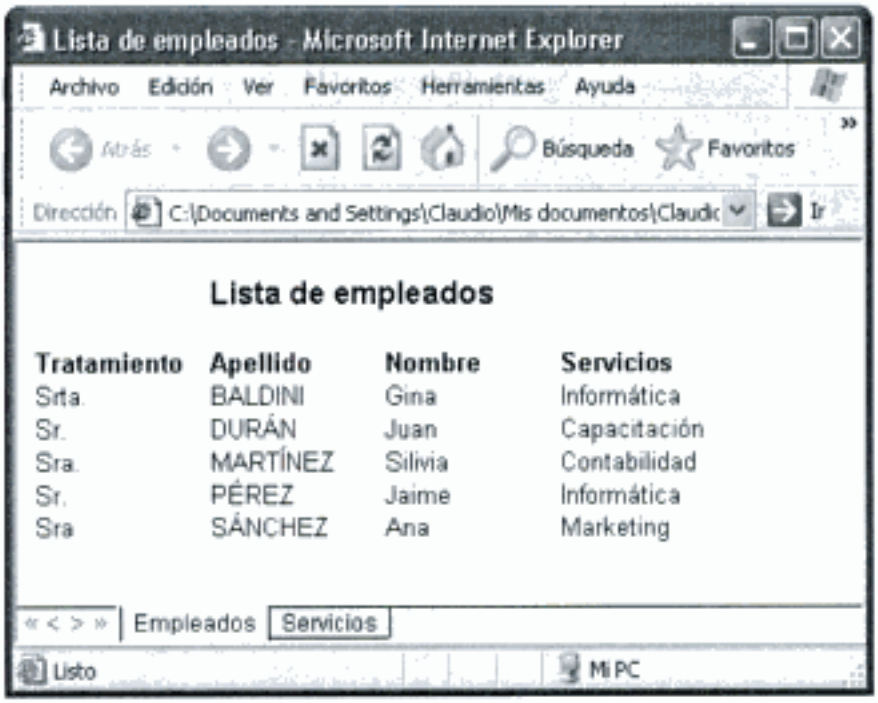
3. Ejemplo

Publicación del libro Empleados: este libro tiene dos hojas de cálculo: Empleados y Servicios.

```
Sub Publicar()
   Dim WebPage As PublishObject
   '    Crea un objeto para guardar una página Web
   Set WebPage = ActiveWorkbook.PublishObjects.Add _
       (xlSourceWorkbook, ThisWorkbook.Path & "\Empleados.html" _
       , , , xlHtmlStatic, , "LISTA DE EMPLEADOS")

   '    Publica la página
   With WebPage
      .Publish (True)
   End With
End Sub
```


Copyrighted Material

Vista previa de la página Web Empleados.html creada.



D. Los objetos WebOptions y DefaultWebOptions

Los objetos **WebOptions** y **DefaultWebOptions** contienen los atributos usados por Excel al grabar un documento como página Web.

El objeto **DefaultWebOptions** contiene las opciones Web por defecto de la aplicación Excel: su contenedor es el objeto **Application**. Las propiedades del objeto corresponden a los atributos accesibles al hacer clic en el botón **Office**, **Opciones de Excel** - **Avanzadas** y, en la sección **General**, en **Opciones Web**.

El objeto **WebOptions** contiene las opciones Web del libro especificado: su contenedor es el objeto **Workbook**. Las propiedades del objeto corresponden a los atributos accesibles desde el botón **Herramientas/Opciones Web** al guardar un libro en formato "Página Web".

© Editions ENI - All rights reserved

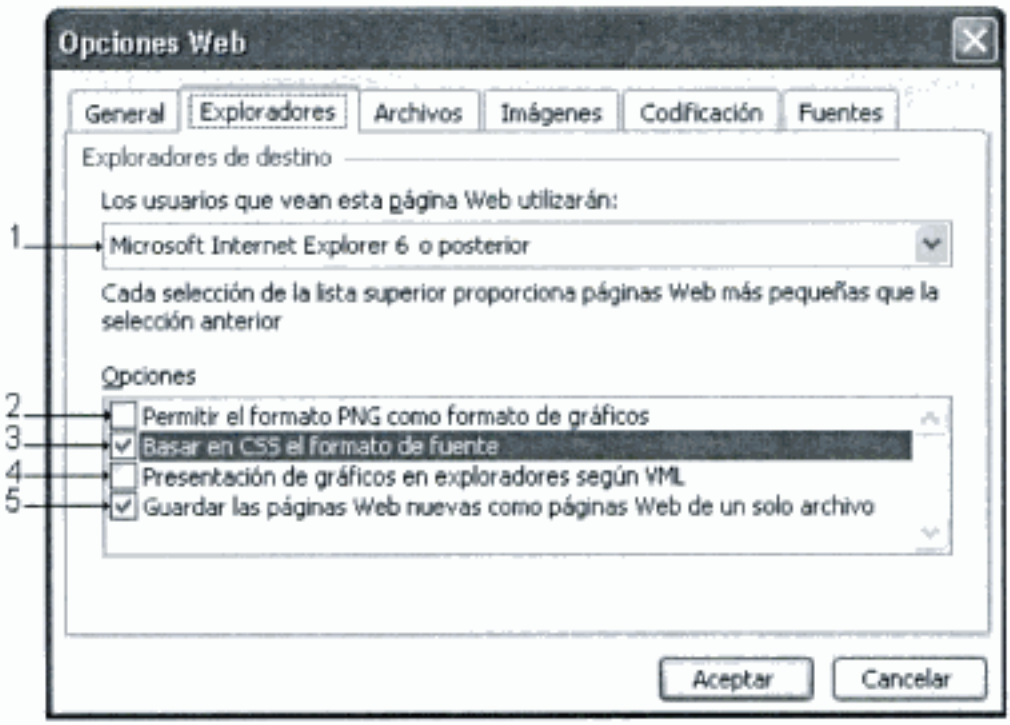
Copyrighted material

Hidden page

Propiedades del objeto DefaultWebOptions

1. SaveHiddenData Boolean
2. LoadPictures Boolean

b. Opciones de la ficha Exploradores



Propiedades comunes a ambos objetos

1. TargetBrowser Constante
   Constantes **msoTargetBrowserIE4 (IE5 o IE6)**
   msoTargetBrowserV3 (o V4)
2. AllowPNG Boolean
3. RelyOnCSS Boolean
4. RelyOnVML Boolean

Propiedades del objeto DefaultWebOptions

5. SaveNewWebPagesAsWebArchives Boolean

© Editions ENI - All rights reserved

Copyrighted material

Hidden page

FolderSuffix

Cadena de caracteres. Sufijo del archivo usado por Excel al guardar un documento como página Web.

PixelsPerInch

Entero largo. Densidad (cantidad de píxeles por pulgada) de las imágenes y celdas de tablas de una página Web.

ScreenSize

Constante. Tamaño mínimo de pantalla ideal (ancho por alto, en píxeles) que debe utilizarse al ver el documento con un explorador Web.
(ejemplo: msoScreenSize 800 x 600, msoScreenSize 1024 x 768, etc.).

Propiedades del objeto DefaultWebOptions

AlwaysSaveInDefaultEncoding

Boolean. Indica si se usa la codificación predeterminada al guardar una página Web.

2. Método del objeto WebOptions

UseDefaultFolderSuffix

Establece el sufijo de carpeta para el libro especificado, correspondiente al idioma que haya seleccionado o instalado.

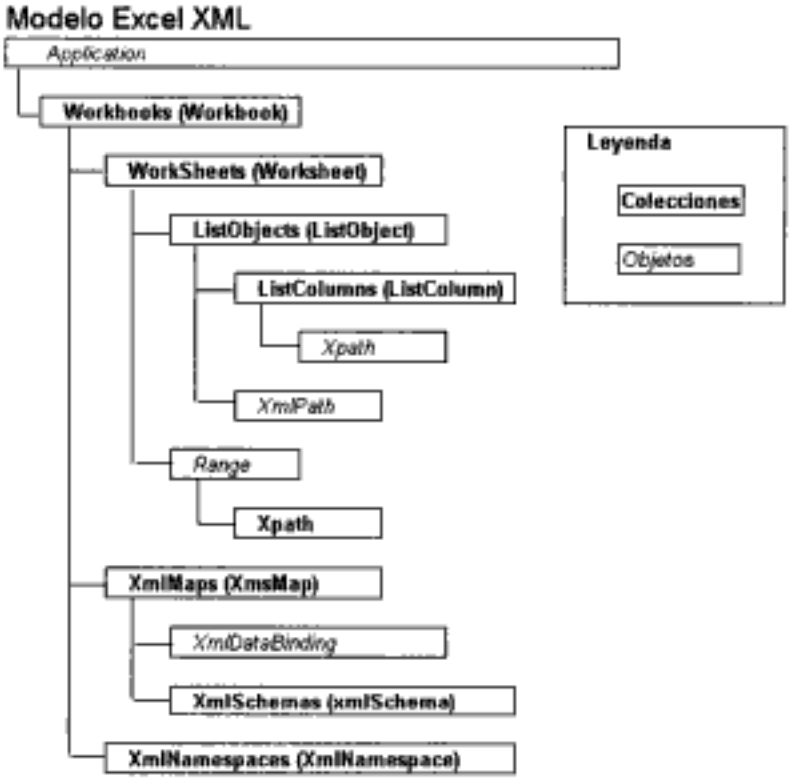
E. Importación, exportación y asignación de archivos XML

En la versión 2007 de Excel, pueden relacionarse datos en formato XML con las celdas o listas de datos Excel.

El Modelo de Objeto Excel XML presentado aquí describe los nuevos objetos que permiten cargar datos en formato XML en los libros Excel.

© Editions ENI - All rights reserved

Copyrighted material



1. Colecciones

ListObjects

Colección de las listas de una hoja de cálculo Excel. Estas listas pueden contener datos XML.

ListColumns

Colección de las columnas de una lista Excel.

XmlMaps

Colección de los objetos **XmlMap** de un libro. Estos objetos se usan para controlar la relación entre los rangos de celdas Excel y los elementos de un esquema XML.

XmlSchemas

Colección de los esquemas XML contenidos en un objeto XML.

XmlNamespaces

Colección de los espacios de nombres XML incluidos en el libro especificado.

Copyrighted Material

2. Métodos del objeto Workbook

XmlImport

Importa un XMLMap a un archivo XML.

Ejemplo:

```
Sub ImportXML()
Dim MapClients As XmlMap

'   Importa el archivo clientes.xml en la hoja activa
ActiveWorkbook.XmlImport Url:=ActiveWorkbook.Path _
     & "\Clientes.xml", ImportMap:=MapClients, _
     Overwrite:=True, Destination:=Range("A1")
MapClients.Name = "Clientes"
End Sub
```

SaveAsXMLData

Permite exportar los datos de un XMLMap a un archivo XML.

Ejemplo:

```
Sub ExportXML()

'   Exporta el XMLMap al archivo Clientes2.xml
ActiveWorkbook.SaveAsXMLData _
       Filename:=ActiveWorkbook.Path & "\Clientes2.xml", _
       Map:=ActiveWorkbook.XmlMaps(1)
End Sub
```

3. Eventos del objeto Workbook

AfterXmlExport

Ocurre tras guardar o exportar Microsoft Excel datos del libro especificado a un archivo de datos XML.

AfterXmlImport

Ocurre tras actualizarse una conexión de datos XML existente o importar nuevos datos XML al libro especificado de Microsoft Excel.

© Editions ENI - All rights reserved

Copyrighted material

BeforeXmlExport

Ocurre antes de guardar o exportar datos del libro especificado a un archivo de datos XML.

BeforeXmlImport

Ocurre antes de actualizar una conexión de datos XML existente o antes de importar nuevos datos XML a un libro de Microsoft Excel.

Ejemplo:

```
Private Sub Workbook_AfterXmlImport(ByVal Map As XmlMap, _
        ByVal IsRefresh As Boolean, ByVal Result As XlXmlImportResult)
If Result = xlXmlImportSuccess Then
   MsgBox "Importación exitosa"
Else
   MsgBox "Problema con la importación del archivo " & Map.Name
End If
End Sub
```

```
Private Sub Workbook_BeforeXmlImport(ByVal Map As XmlMap, _
     ByVal Url As String, ByVal IsRefresh As Boolean, Cancel As Boolean)
If MsgBox("Desea importar el archivo " & Url & " ? ", _
   vbQuestion + vbYesNo) = vbNo Then
       MsgBox "Importación cancelada", vbExclamation
       Cancel = True
End If
End Sub
```

4. Métodos del objeto XmlMap

Delete

Permite quitar un XMLMap.
Ejemplo: ActiveWorkbook.XmlMaps(1).Delete

Import

Importa los datos del archivo de datos XML especificado a las celdas asignadas al objeto **XmlMap** especificado.
Ejemplo: ActiveWorkbook.XmlMaps(1).Import "C:\Empleados.xml"

Copyrighted Material

Export

Exporta a un archivo de datos XML el contenido de las celdas asignadas al objeto **XmlMap** especificado.
Ejemplo: ActiveWorkbook.XmlMaps(1).Export "C:\Empleados.xml"

F. El objeto HyperLink

El objeto **HyperLink** representa un **hipervínculo** contenido en una hoja de cálculo, un rango de celdas o un gráfico.

El objeto **Hyperlink** pertenece a la colección **Hyperlinks** de los objetos contenedores **Range**, **Workbook** y **Chart**.

1. Propiedades

Address

Cadena de caracteres. Dirección de la celda que contiene el hipervínculo especificado.

EmailSubject

Cadena de caracteres. Texto de la línea de asunto de correo electrónico del hipervínculo especificado (propiedad usada con los hipervínculos de los mensajes de correo electrónico).

Name

Cadena de caracteres. Nombre del hipervínculo.

Range

Objeto **Range**. Rango vinculado al hipervínculo.

ScreenTip

Cadena de caracteres. Texto de la etiqueta informativa del hipervínculo especificado.

Shape

Objeto **Shape**. Forma vinculada al hipervínculo especificado.

© Editions ENI - All rights reserved

Copyrighted Material

SubAddress

Cadena de caracteres. Ubicación dentro del documento a la que hace referencia el hipervínculo.

TextToDisplay

Cadena de caracteres. Texto que se mostrará para el hipervínculo especificado.

Type

Entero largo. Tipo del hipervínculo especificado.

2. Métodos

AddToFavorites

Agrega un acceso directo al hipervínculo en la carpeta Favoritos.

CreateNewDocument

Crea un nuevo documento vinculado al hipervínculo especificado.

Delete

Elimina el hipervínculo especificado.

Follow

Carga el documento de destino asociado al hipervínculo especificado y muestra el documento en la aplicación apropiada.

Copyrighted Material

Capítulo 12: Programación Windows

A. Presentación de las API 326

B. Llamar a una función de la API Windows 327

1. Sintaxis de la instrucción Declare 327

2. Paso de argumentos 328

C. Lista de funciones API Windows 328

D. Ejemplos de uso de funciones API Windows 329

1. Recuperar el directorio Windows 329

2. Abrir la calculadora de Windows 330

E. El objeto FileSystemObject 332

1. Métodos 332

2. Propiedades 333

3. Ejemplo: copia de archivos Excel 333

Copyrighted material

A. Presentación de las API

La interfaz de programación Windows **API** (*Application Programming Interface*) ofrece funciones que le permiten controlar los más ínfimos aspectos del sistema. Usted puede extender y personalizar sus aplicaciones Excel llamando a funciones Windows API desde VBA. Si bien Excel continúa evolucionando y su lenguaje de programación nativo (VBA) integra cada vez más funciones de sistema, para ciertas tareas se deben usar las funciones API.

Una API es una serie de funciones que se pueden usar para trabajar con un componente, una aplicación o un sistema operativo. Se compone generalmente de uno o más archivos DLL (*Dynamic Link Library* o biblioteca de vínculos dinámicos).

La API más usada es la **API Windows** que incluye las DLL que forman el sistema operativo Windows. Cada aplicación Windows interactúa directa o indirectamente con la API Windows. Esto garantiza un comportamiento coherente de todas las aplicaciones que funcionan bajo Windows.

Las DLL de la aplicación Windows más usadas son las siguientes:

Kernel32.dll funciones de bajo nivel del sistema operativo, tales como la administración de memoria y de recursos,

User32.dll funciones de administración Windows, tales como el tratamiento de mensajes, relojes, menús y comunicación,

GDI32.dll biblioteca GDI (*Graphics Device Interface*) que contiene las funciones de salida hacia los periféricos (gráficos, contexto de visualización y administración de fuentes).

Existen otras API, como por ejemplo la interfaz **MAPI** (*Mail Application Programming Interface*) que permite escribir **aplicaciones de correo electrónico**.

Copyrighted Material

Hidden page

`nom_alias` Nombre del procedimiento o de la función en la DLL.

`lista_argumentos` [Optional][ByVal | ByRef][ParamArray] nomvariable[()][As type] (ver capítulo 2: El lenguaje VBA).

Ciertas DLL no proveen un nombre para sus procedimientos y funciones, sino un número ordinal. La declaración de estos procedimientos o funciones usa la misma sintaxis, pero se debe definir el número ordinal a nivel del alias con un carácter numeral (#) seguido del número (ejemplo: Alias "#52").

2. Paso de argumentos

Las funciones y procedimientos de las DLL están escritos principalmente en lenguaje C y hacen, por lo tanto, referencia a su sintaxis. Por eso, el paso de argumentos a un procedimiento o una función de una DLL desde Visual Basic no siempre es simple. A nivel de las DLL que usan la sintaxis de lenguaje C, todos los argumentos se pasan por valor, salvo las matrices.
Las cadenas en lenguaje C se consideran como matrices de caracteres.
Ciertos argumentos de procedimientos de DLL pueden aceptar distintos tipos de datos (a semejanza de los Variant); deben ser declarados como tipo Any (ejemplo: `variable As Any`). Para este tipo de argumento, Visual Basic considera que él mismo se pasa sistemáticamente por referencia; si hay que pasarlo por valor, se lo debe explicitar a nivel del llamado (y no de la declaración) del procedimiento o de la función con la mención `ByVal`.

Por defecto, Visual Basic pasa los argumentos por referencia.

C. Lista de funciones API Windows

Esta lista incluye las funciones de la API Windows habitualmente usadas. Los ejemplos de uso de algunas de estas funciones se escriben en el siguiente párrafo.

GetWindowsDirectory()

Devuelve la ruta completa del directorio de Windows.

GetSystemDirectory()

Devuelve la ruta completa del directorio de sistema de Windows.

Copyrighted material

Hidden page

Llamado de la función API:

```
Function GetWinPath() As String

' Esta función VBA devuelve el directorio Windows
Dim StrResult As String
Dim StrProfile As String

StrResult = String(255, " ")
StrProfile = GetWindowsDirectory(StrResult, 255)

  ' Corta la cadena al primer carácter nulo
If StrProfile <> "" Then
   StrResult = Trim(StrResult)
   GetWinPath = Left(StrResult, InStr(1, StrResult, vbNullChar) - 1)
Else
   GetWinPath = ""
End If

End Function
```

2. Abrir la calculadora de Windows

Este ejemplo prueba si la calculadora Windows está activa, y la inicia si no lo está.

Aquí se usan dos funciones API: la primera, **FindWindow**, busca la ventana "Calculadora"; la segunda, **FindExecutable**, busca la ubicación del archivo Calc.exe.

Declaración de la función API:

```
'   Esta función API busca una ventana
Public Declare Function FindWindow Lib "user32" Alias _ "FindWindowA"
(ByVal lpClassName As String, ByVal  _
lpWindowName As String) As Long

'   Esta función API busca un archivo ejecutable
Public Declare Function FindExecutable Lib "shell32.dll" _
Alias "FindExecutableA" (ByVal lpFile As String, _
ByVal lpDirectory As String, ByVal lpResult As String) As Long
```


Copyrighted material

Hidden page

E. El objeto FileSystemObject

El objeto **FileSystemObject** proporciona acceso al **sistema de archivos de un equipo**. Permite especialmente buscar, crear, eliminar, mover archivos o carpetas.

1. Métodos

Métodos relativos a los archivos

CopyFile

Copia uno o más archivos de un lugar a otro.

CreateTextFile

Crea un nombre de archivo especificado y devuelve un objeto TextStream que se puede utilizar para leer o escribir en un archivo.

DeleteFile

Elimina un archivo especificado.

FileExists

Devuelve un valor Boolean que indica si existe el archivo especificado.

MoveFile

Mueve uno o más archivos de un lugar a otro.

OpenTextFile

Abre el archivo especificado y devuelve un objeto TextStream que se puede utilizar para leer el archivo o agregar datos.

Métodos relativos a las carpetas

CopyFolder

Copia una carpeta de un lugar a otro.

CreateFolder

Crea una carpeta.

Copyrighted Material

DeleteFolder

Elimina una carpeta especificada junto con su contenido.

FolderExists

Devuelve un valor Boolean que indica si existe la carpeta especificada.

MoveFolder

Mueve una o más carpetas de un lugar a otro.

Métodos relativos a las unidades de disco

DriveExists

Devuelve un valor Boolean que indica si existe la unidad especificada.

GetDrive

Devuelve un objeto **Drive** que corresponde a la unidad de la ruta especificada.

GetDriveName

Devuelve una cadena que contiene el nombre de la unidad para una ruta especificada.

2. Propiedades

Drives

Devuelve una colección formada por los objetos **Drive** disponibles en el equipo local.

3. Ejemplo: copia de archivos Excel

El siguiente ejemplo busca todos los archivos Excel que hay en la misma carpeta que el libro activo y los copia en el directorio "C:\Archivos Excel".

```
Sub CopiaArchivos()
Dim fso As Object
Dim strFile As String
'   Crea un nuevo directorio (salvo que ya exista)
Set fso = CreateObject("Scripting.FileSystemObject")            .../...
```

© Editions ENI - All rights reserved

Copyrighted material

```
.../...
If Not fso.folderExists("C:\Archivos Excel") Then
   fso.createfolder ("C:\Archivos Excel")
End If

'   Busca los archivos Excel y los copia en el directorio
strFile = Dir(ThisWorkbook.Path & "\*.xls*", vbNormal)
Do While strFile <> ""
    fso.copyfile ThisWorkbook.Path & "\" & strFile, "C:\Archivos Excel\"
    '   Archivo siguiente
    strFile = Dir
Loop
End Sub
```


Copyrighted material

Capítulo 13: Código de una miniaplicación

A. Presentación general 336

B. Descripción de la aplicación GestPresupuesto 337

C. Módulo ThisWorbook 338

1. Presentación 338
2. Código VBA 339

D. Formulario NuevoPresupuesto 341

1. Presentación 341
2. Lista de controles 342
3. Lista de celdas con nombre del modelo Presupuesto.xltx . . 343
4. Código VBA 343

E. Formulario BuscarPresupuesto 346

1. Presentación 346
2. Lista de controles 346
3. Código VBA 347

F. Módulos ProcActions y ProcGene 350

1. Código VBA del módulo ProcActions 350
2. Código VBA del módulo ProcGene 351

Copyrighted material

A. Presentación general

La aplicación Excel presentada en este capítulo permite administrar presupuestos realizados con Excel.

Las principales funcionalidades de esta aplicación son:

- creación de un nuevo presupuesto a partir de un modelo,
- búsqueda de presupuestos en función de criterios (cliente, fecha), con la posibilidad de abrir o de eliminar uno o más presupuestos,
- creación de nuevos clientes y búsqueda de clientes.

Todos los archivos necesarios para la aplicación deben instalarse en el mismo directorio. Estos archivos se entregan con los ejemplos del libro y son los siguientes:

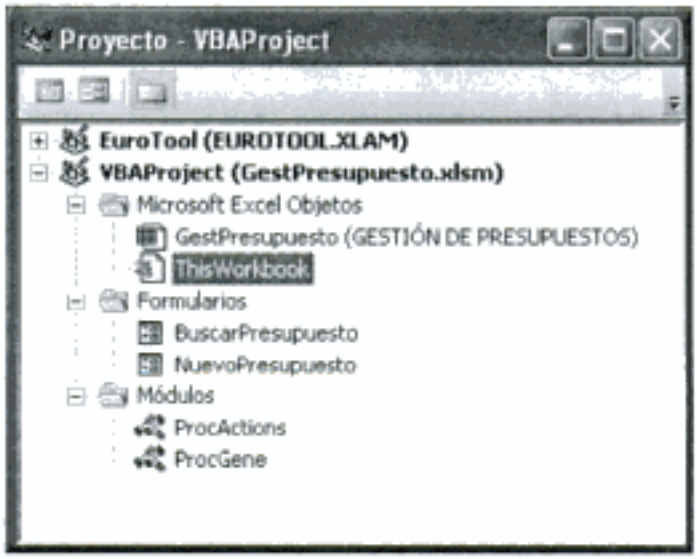
- El archivo que contiene el código de la aplicación Excel: GestPresupuesto.xlsm,
- La base de datos Access que incluye la tabla de clientes y los formularios de búsqueda y de creación de un cliente: Presupuesto.accdb,
- El modelo Excel que sirve de base a la generación de presupuestos: Presupuesto.xltx.

Los presupuestos generados se presentan como archivos Excel denominados de la siguiente manera: fecha de creación en la forma AAAAMMDD y extensión xlsx (ejemplo: 20070410.xlsx). Éstos se generan en un subdirectorio de la aplicación que lleva el nombre del código de cliente (ejemplo: ALFKI, ANATR, etc.).

Copyrighted Material

Hidden page

- dos formularios **BuscarPresupuesto** y **NuevoPresupuesto** permiten, respectivamente, buscar y crear un nuevo presupuesto.
- dos módulos estándares: el módulo **ProcActions** contiene los procedimientos llamados por los comandos de menú específicos de la aplicación (estos procedimientos están asociados a las propiedades onAction de los diferentes comandos del menú); el módulo **ProcGene** contiene las variables públicas y los procedimientos generales de la aplicación.



C. Módulo ThisWorbook

1. Presentación

Este módulo permite:

- agregar los comandos de menús en la ficha **Complementos**,
- modificar la presentación de Excel: oculta la barra de fórmulas y los encabezados de filas y columnas,
- restablecer el entorno Excel cuando se desactiva el libro: muestra la barra de fórmulas y los encabezados de filas y columnas,
- suprimir los comandos de menús específicos de la aplicación cuando se cierra el libro.

Copyrighted Material

Hidden page

```
.../...
        On Error Resume Next
        m_Ctrl.Delete
    End If
Next m_Ctrl
On Error GoTo 0
End Sub
```

```
Private Sub Workbook_Deactivate()
Dim m_Ctrl As CommandBarControl
'   Muestra la barra de fórmulas y los encabezados
With Application
    .DisplayFormulaBar = True
    .ActiveWindow.DisplayHeadings = True
End With
End Sub
```

```
Private Sub Workbook_Open()
Dim cmd As CommandBar
Dim opt As CommandBarControl
'   Ruta de acceso de la aplicación
strFolder = ThisWorkbook.Path & "\"
'   Agrega los comandos personalizados
'   a nivel de la ficha Complementos
Set cmd = Application.CommandBars("Tools")
'   Nuevo Presupuesto
Set opt = cmd.Controls.Add(Type:=msoControlButton, _
              ID:=18, Before:=1)
opt.Caption = "Nuevo presupuesto"
opt.OnAction = "Crear_Presupuesto"
opt.TooltipText = "Crea un nuevo presupuesto"
'   Buscar Presupuesto
Set opt = cmd.Controls.Add(Type:=msoControlButton, _
              ID:=1849, Before:=2)
opt.OnAction = "Buscar_Presupuesto"
opt.Caption = "Buscar presupuesto  "
opt.TooltipText = "Busca un presupuesto por cliente y por fecha"
'   Opción inactiva
Set opt = cmd.Controls.Add(Type:=msoControlButton, _
               Before:=3)
opt.OnAction = " "
opt.Caption = " "
opt.Enabled = False                                        .../...
```


Copyrighted Material

```
.../...
'   Nuevo Cliente
Set opt = cmd.Controls.Add(Type:=msoControlButton, _
              ID:=18, Before:=1)
opt.Caption = "Nuevo cliente"
opt.OnAction = "Agrega_Cliente"
opt.TooltipText = "Crea un nuevo cliente"

'   Buscar Cliente
Set opt = cmd.Controls.Add(Type:=msoControlButton, _
              ID:=1849, Before:=2)
opt.OnAction = "Buscar_Cliente"
opt.Caption = "Buscar cliente"
opt.TooltipText = "Busca un cliente"
End Sub
```

D. Formulario NuevoPresupuesto

1. Presentación

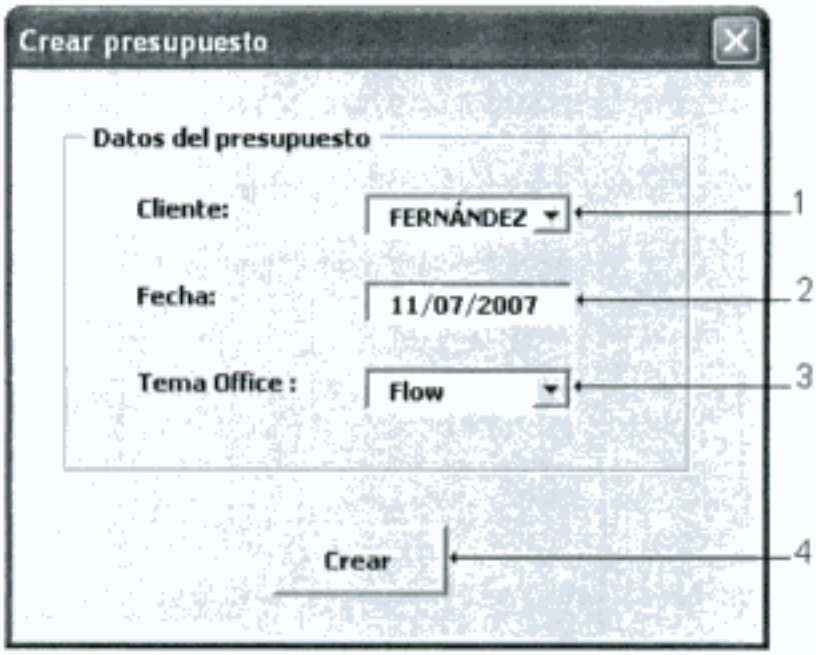
Este módulo permite:

- crear un nuevo presupuesto a partir del modelo Presupuesto.xltx, y guardar el presupuesto en el subdirectorio del cliente,
- mostrar los datos del cliente en las celdas con nombre (CodCli, Empresa, Dirección, etc.) del libro,
- aplicar al libro el tema seleccionado en el formulario.

© Editions ENI - All rights reserved

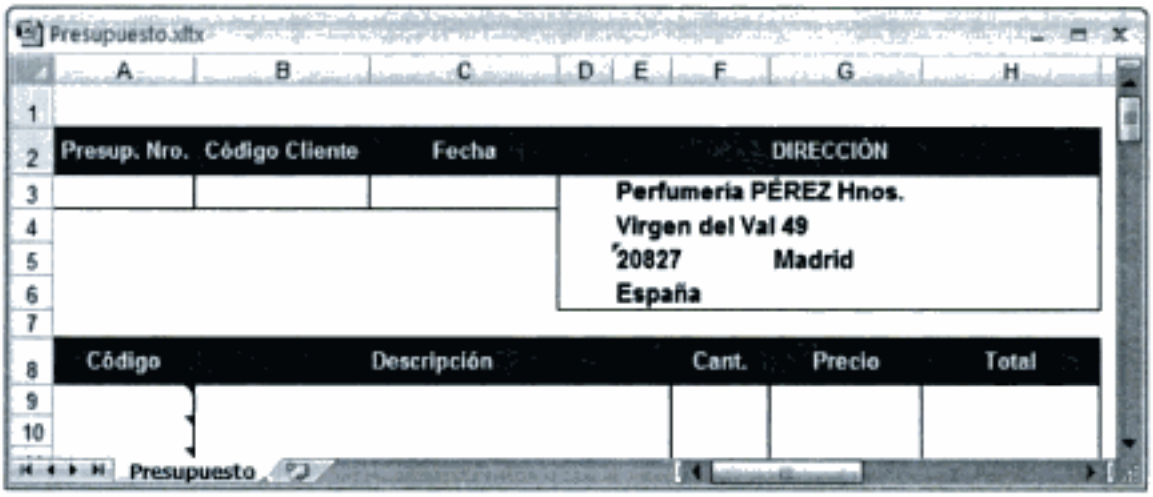
Copyrighted material

2. Lista de controles



Nombre del control		Descripción
1)	cboClient	Cuadro de lista desplegable
2)	txtFecha	Cuadro de texto
3)	cboTema	Cuadro de lista desplegable
4)	cmdCrear	Botón de comando

3. Lista de celdas con nombre del modelo Presupuesto.xltx



1. Presupuesto
2. CodCli
3. Fecha
4. Empresa
5. Direccion
6. CPostal
7. Ciudad
8. Pais

4. Código VBA

```
Option Explicit

Private Sub UserForm_Initialize()
'   Muestra la lista de clientes
Lista_Clientes ("NuevoPresupuesto")
'   Muestra los temas Microsoft Office
Mostrar_Temas_Office
'   Fecha por defecto
txtFecha = Format(Date, "DD/MM/YYYY")
End Sub

Private Sub txtFecha_BeforeUpdate(ByVal Cancel As _
    MSForms.ReturnBoolean)
'   Controla la fecha introducida
If txtFecha <> "" Then Cancel = Not Ctrl_Fecha(txtFecha)

End Sub                                                    .../...
```

© Editions ENI - All rights reserved

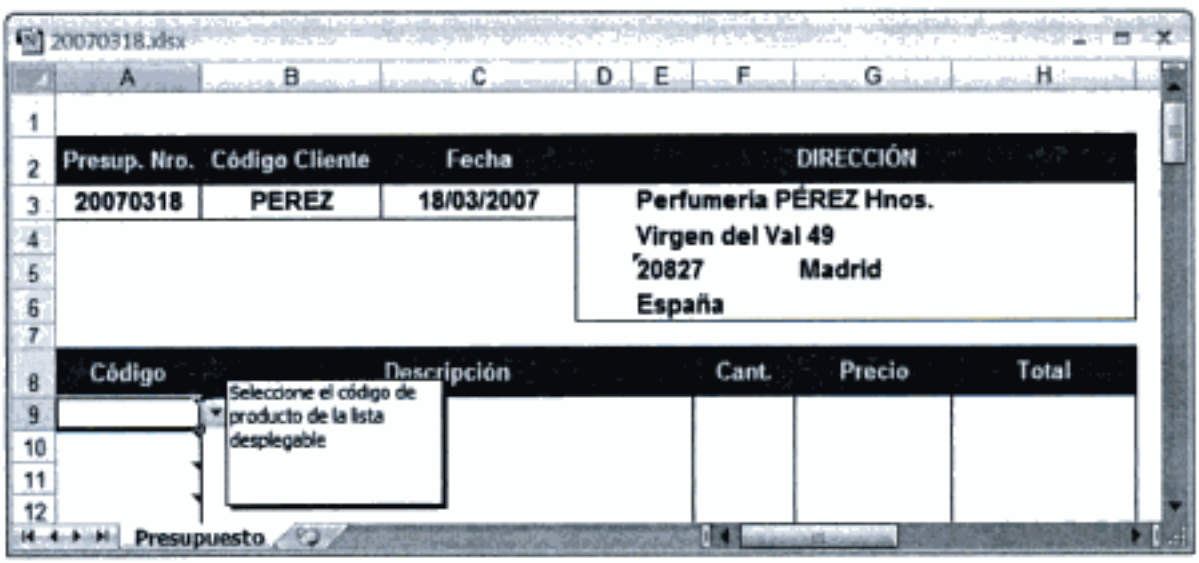
Copyrighted material

```
.../...
Private Sub Mostrar_Temas_Office()
Dim strPath As String
Dim strfile As String
'   Muestra la lista de temas Microsoft Office
cboTema.Clear
'   Ruta de acceso a las plantillas
strPath = Left(Application.Path, Len(Application.Path) - 9) _
        & "\Document Themes 12\"
strfile = Dir(strPath & "*.thmx")
'   Muestra el nombre de archivo sin la extensión
Do While strfile <> ""
   cboTema.AddItem Left(strfile, Len(strfile) - 5)
   strfile = Dir
Loop
End Sub
Private Sub CmdCrear_Click()
Dim wbk As Workbook
Dim wbkName As String
Dim fso As Object
Dim strTheme As String
'   Controla los datos introducidos
If cboClient = "" Or txtFecha = "" Then
   MsgBox "Cliente y fecha obligatorios", vbExclamation
   Exit Sub
End If
'   Cierra los libros (excepto ThisWorkbook)
For Each wbk In Workbooks
   If wbk.Name <> ThisWorkbook.Name Then
      wbk.Close
   End If
Next wbk
'   Verifica que el libro ya no exista
wbkName = strFolder & cboClient & "\" & Right(txtFecha, 4) & _
          Mid(txtFecha, 4, 2) & Left(txtFecha, 2) & ".xlsx"
If Dir(wbkName) <> "" Then
   MsgBox "El libro " & wbkName & " ya existe", vbExclamation
   Exit Sub
End If
'   Abre un nuevo libro basado en la plantilla Presupuesto.xltx
Set wbk = Workbooks.Add(Template:=strFolder & "Presupuesto.xltx")
.../...
```


Copyrighted material

```
.../...
'   Crea el subdirectorio del cliente si no existe
If Dir(strFolder & cboClient, vbDirectory) = "" Then
   Set fso = CreateObject("Scripting.FileSystemObject")
   fso.createfolder (strFolder & cboClient)
End If
'   Guarda el libro en el directorio
wbkName = Right(txtFecha, 4) & Mid(txtFecha, 4, 2) & Left(txtFecha, 2) _
                                                    & ".xlsx"
wbk.SaveAs strFolder & cboClient & "\" & wbkName
wbk.Activate
'   Asigna las celdas del libro a partir de la tabla Cliente
'   (Procedimiento del módulo ProcGene)
Mostrar_Cliente wbk, cboClient
wbk.ActiveSheet.Range("A9").Activate
'   Aplica el tema Office seleccionado
strTheme = Left(Application.Path, Len(Application.Path) - 9) _
        & "\Document Themes 12\" & cboTema & ".thmx"
wbk.ApplyTheme strTema
'   Cierra el formulario
Unload Me
End Sub
```

Después de ejecutar este módulo, se habrá creado un nuevo presupuesto y se muestra en Excel para que el usuario lo complete:



© Editions ENI - All rights reserved

Copyrighted material

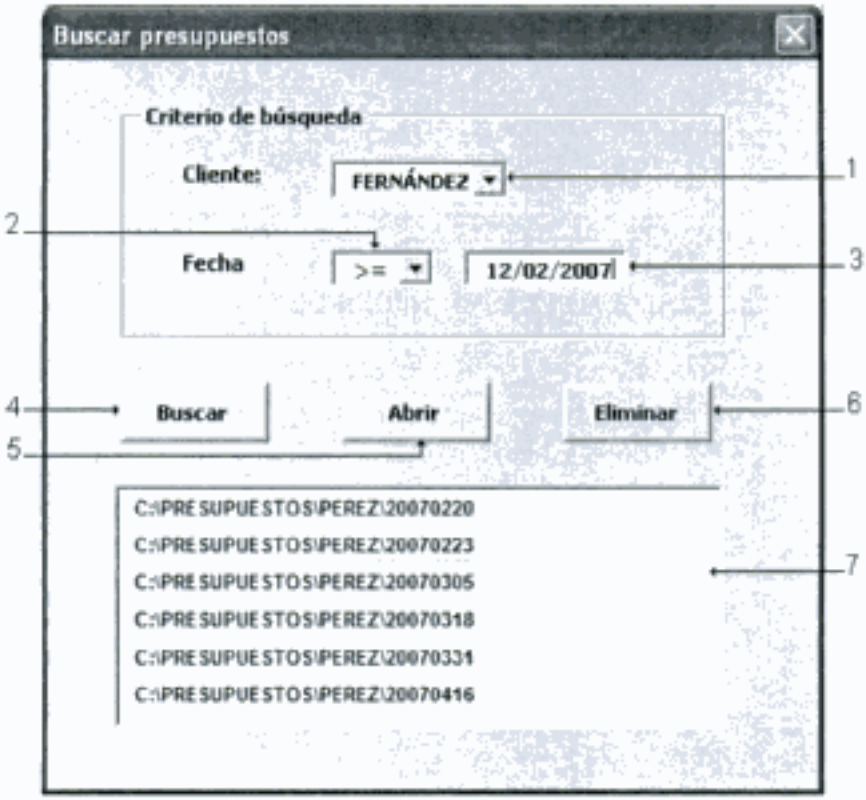
E. Formulario BuscarPresupuesto

1. Presentación

Este módulo permite:

- buscar presupuestos en función del código de cliente y/o la fecha del presupuesto (si no se indica ningún criterio, se muestran todos los presupuestos disponibles),
- abrir o eliminar uno o más presupuestos en la lista de presupuestos obtenida.

2. Lista de controles



Nombre del control	Descripción
1. cboClient	Cuadro de lista desplegable modificable.
2. cboOpe	Cuadro de lista desplegable modificable que contiene los operadores de comparación (>= o <=).

3. txtFecha Cuadro de texto.
4. cmdBuscar Botón de comando.
5. cmdSupr Botón de comando.
6. cmdAbrir Botón de comando.
7. lstPresup Cuadro de lista desplegable.

3. Código VBA

```
Option Explicit
_____________________________________________________________
Private Sub UserForm_Initialize()
'   Operador para el campo Fecha
cboOpe.AddItem ">="
cboOpe.AddItem "<="
'   Selecciona varios presupuestos
lstPresup.MultiSelect = fmMultiSelectMulti
'   Muestra los clientes en la lista desplegable
Lista_Clientes ("BuscarPresupuesto")
End Sub
_____________________________________________________________
Private Sub txtFecha_BeforeUpdate _
        (ByVal Cancel As MSForms.ReturnBoolean)
'   Controla la fecha introducida
If txtFecha <> "" Then Cancel = Not Ctrl_Fecha(txtFecha)
End Sub
_____________________________________________________________
Private Sub CmdAbrir_Click()
'   Abre los presupuestos seleccionados
On Error GoTo Err:
For j = 0 To lstPresup.ListCount - 1
   If lstPresup.Selected(j) Then
      Workbooks.Open lstPresup.List(j)
   End If
Next j
On Error GoTo 0
Unload Me
Exit Sub

Err:
'   No se pueden abrir dos presupuestos con el mismo nombre
'   en la misma aplicación Excel
If Err.Number = 1004 Then
   MsgBox Err.Description
End If                                                    .../...
```

© Editions ENI - All rights reserved

Copyrighted material

Hidden page

```
.../...
Private Function Ctrl_Presup(strFileName As String) As Boolean
Dim dte
'   Controla la fecha del presupuesto
Ctrl_Presup = False
If cboOpe <> "" And txtFecha <> "" Then
   strFileName = Left(strFileName, Len(strFileName) - 5)
   dte = Right(strFileName, 2) & "/" _
       & Mid(strFileName, 5, 2) & "/" & Left(strFileName, 4)
   If Not IsDate(dte) Then Exit Function
      If cboOpe = ">=" And DateValue(dte) < DateValue(txtFecha) _
         Then Exit Function
      If cboOpe = "<=" And DateValue(dte) > DateValue(txtFecha) _
         Then Exit Function
End If
Ctrl_Presup = True
End Function
Private Sub cmdSupr_Click()
Dim strLista As String
Dim fso As Object
'   Muestra los presupuestos seleccionados
For i = 0 To lstPresup.ListCount - 1
   If lstPresup.Selected(i) Then
      strLista = strLista & vbCr & lstPresup.List(i)
   End If
Next i
'   Elimina los presupuestos seleccionados después de pedir confirmación
If MsgBox("¿Desea eliminar estos presupuesto? " & strLista, _
   vbQuestion & vbYesNo) = vbYes Then
   Set fso = CreateObject("Scripting.FileSystemObject")
   For i = 0 To lstPresup.ListCount - 1
      If lstPresup.Selected(i) Then
         fso.Deletefile lstPresup.List(i)
      End If
   Next i
End If
'   Actualiza la lista de presupuestos
cmdBuscar_Click
End Sub
```

© Editions ENI - All rights reserved

Copyrighted material

Hidden page

```
.../...
AccessActivo = True
Exit Function
Err:
blnAccess = False
GoTo ejecutaAccess:
Exit Function
Err2:
AccessActivo = False
End Function

Sub Crear_Presupuesto()
'   Muestra el formulario NuevoPresupuesto
NuevoPresupuesto.Show
End Sub

Sub Buscar_Presupuesto()
'   Muestra el formulario de búsqueda de presupuestos
BuscaPresup.Show
End Sub
```

```
Sub Salir()
'   Pide confirmar la salida de la aplicación
If MsgBox("¿Desea salir de la aplicación Presupuesto?", _
   vbQuestion & vbYesNo) = vbYes Then
   On Error Resume Next
   appAccess.Quit
   On Error GoTo 0:
   ActiveWorkbook.Close
End If
End Sub
```

2. Código VBA del módulo ProcGene

```
Option Explicit
'   Variables públicas
Public i As Integer
Public j As Integer
'   Directorio de la aplicación
Public Const strFolder = "C:\Presupuesto\"
'   Objetos ADO
Private cnnCli As ADODB.Connection
Private rstCli As ADODB.Recordset                    .../...
```

© Editions ENI - All rights reserved

Copyrighted material

Hidden page

```
.../...
'   Muestra los datos del cliente en las celdas
'   del libro Presupuesto
If Not rstCli.EOF Then
   With wbk.Sheets(1)
      .Range("FECHA") = NuevoPresup.txtFecha
      .Range("PRESUPUESTO") = Left(wbk.Name, Len(wbk.Name) - 4)
      .Range("CODCLI") = rstCli("CLI_CODCLI")
      .Range("EMPRESA") = rstCli("CLI_EMPRESA")
      .Range("DIRECCIÓN") = rstCli("CLI_DIRECCIÓN")
      .Range("CPOSTAL") = rstCli("CLI_CPOSTAL")
      .Range("CIUDAD") = rstCli("CLI_CIUDAD")
      .Range("PAÍS") = UCase(rstCli("CLI_PAÍS"))
   End With
End If
'   Cierra los objetos Access
rstCli.Close
cnnCli.Close
Set rstCli = Nothing
Set cnnCli = Nothing
End Sub
```

```
Public Function AbreBase() As Boolean
'   Abre la base Presupuesto.mdb
On Error GoTo Err:
Set cnnCli = New ADODB.Connection
With cnnCli
   .Provider = "Microsoft.JET.OLEDB.4.0"
   .Open strFolder & "Presupuesto.mdb"
End With
On Error GoTo 0
AbreBase = True
Exit Function
Err:
On Error GoTo 0
AbreBase = False

MsgBox "Problema al abrir la base Presupuesto.mdb", _
vbExclamation
End Function
```

```
Function Ctrl_Fecha(zDate As Control) As Boolean
Dim m_sDate As String
Dim m_date As Date
Dim x As Integer                                        .../...
```

© Editions ENI - All rights reserved

Copyrighted material

Hidden page

Anexos

A. Lista de instrucciones 357

1. Fecha Hora/Matemáticas 357
2. Declaración 357
3. Error 360
4. Archivo 360
5. Estructuración 361
6. Sistema 363
7. Diversas 364

B. Lista de funciones 365

1. Conversiones 365
2. Cadenas de caracteres 367
3. Matemáticas 370
4. Financieras 372
5. Fechas y horas 373
6. Archivos, Sistema 376
7. Verificación de variables 377
8. Interacción 378
9. Matrices 379
10. SQL 380

Copyrighted material

11. Diversas 381
12. Solver 382

C. Constantes VBA 383
1. Constantes de color 383
2. Constantes de fecha 384
3. Constantes de teclas correspondientes a letras y números . 384
4. Constantes de teclas de función 385
5. Constantes de teclas diversas 385

Copyrighted material

A. Lista de instrucciones

LSet

Alinea a la izquierda una cadena de caracteres dentro de una variable tipo cadena.
o
Copia una variable de un tipo definido por el usuario a otra variable de otro tipo definido por el usuario.

Mid

Reemplaza una cantidad especificada de caracteres dentro de una variable cadena por los caracteres extraídos de otra cadena.

RSet

Alinea a la derecha una cadena de caracteres dentro de una variable tipo cadena.

1. Fecha Hora/Matemáticas

Date

Devuelve la fecha del sistema en curso.

Randomize

Inicializa el generador de números aleatorios.

Time

Devuelve la hora del sistema.

2. Declaración

Const

Declara las constantes que hay que utilizar en lugar de valores fijos.

Declare

Se utiliza a nivel de módulo para declarar las referencias a procedimientos externos en una biblioteca DLL o un recurso de código Macintosh.

© Editions ENI - All rights reserved

Copyrighted Material

Hidden page

Private

Declara las variables privadas y reserva su espacio de almacenamiento en la memoria.

Property Get

Declara el nombre, los argumentos y el código de un procedimiento **Property** que permite leer el valor de una propiedad.

Property Let

Declara el nombre, los argumentos y el código de un procedimiento **Property** que le asigna un valor a una propiedad.

Property Set

Declara el nombre, los argumentos y el código de un procedimiento **Property** que asigna una referencia a un objeto.

Public

Declara las variables públicas y les reserva espacio de almacenamiento en la memoria.

ReDim

Dimensiona variables de tipo tabla dinámica y les reserva espacio de almacenamiento en la memoria.

Set

Asigna una referencia a un objeto.

Static

Define las variables estáticas y les reserva espacio de almacenamiento en la memoria.

Sub

Declara el nombre, los argumentos y el código de un procedimiento Sub.

Type

Define un tipo de datos definido por el usuario.

© Editions ENI - All rights reserved

Copyrighted material

3. Error

Error

Simula la ocurrencia de un error.

On Error

Activa una rutina de tratamiento de errores y especifica su ubicación dentro de un procedimiento. También permite desactivar una rutina de tratamiento de errores.

Resume

Restablece la ejecución del código cuando termina una rutina de tratamiento de errores.

4. Archivo

Close

Finaliza las operaciones de entrada y salida en un archivo abierto con la instrucción `Open`.

FileCopy

Copia un archivo.

Get

Lee los datos de un archivo abierto y los guarda en una variable.

Input #

Lee los datos a partir de un archivo secuencial abierto y los asigna a variables.

Kill

Elimina los archivos del disco.

Line Input #

Lee una línea de datos a partir de un archivo secuencial abierto y la asigna a una variable de tipo cadena.

Copyrighted material

Lock...Unlock

Controla el acceso por parte de otros procesos a todo o parte de un archivo abierto mediante la instrucción `Open`.

Open

Permite ejecutar una operación de entrada y salida en un archivo.

Print #

Escribe los datos con formato en un archivo secuencial.

Put

Escribe el contenido de una variable en un archivo de disco.

Reset

Cierra todos los archivos de discos abiertos con la instrucción `Open`.

Seek

Define la posición de la próxima lectura y escritura en un archivo abierto con la instrucción `Open`.

SetAttr

Define los atributos de un archivo.

Width #

Asigna la longitud de la línea de salida a un archivo abierto con la instrucción `Open`.

Write #

Escribe datos en un archivo secuencial.

5. Estructuración

Call

Transfiere el control a un procedimiento **Sub**, **Function**, DLL o a un procedimiento de recursos de código Macintosh.

© Editions ENI - All rights reserved

Copyrighted material

Hidden page

Stop

Interrumpe la ejecución de un procedimiento.

If...Then...
ElseIf...
Else...End If

Permite la ejecución condicional de un grupo de instrucciones según el resultado de una expresión dada.

While...Wend

Ejecuta una serie de instrucciones mientras se cumpla una condición dada.

With

Ejecuta una serie de instrucciones sobre un único objeto o un tipo definido por el usuario.

6. Sistema

Beep

Emite una señal sonora.

ChDir

Cambia el directorio o la carpeta actual.

ChDrive

Cambia la unidad de disco actual.

MkDir

Crea un nuevo directorio o nueva carpeta.

Name

Modifica el nombre de un archivo, de un directorio o de una carpeta.

RmDir

Elimina un directorio o una carpeta existente.

© Editions ENI - All rights reserved

Copyrighted material

Hidden page

B. Lista de funciones

Las funciones cuyo nombre termina con el signo $ devuelven valores en variables de tipo **String** y no de tipo Variant.

1. Conversiones

CBool

Convierte una expresión a datos de tipo Boolean.

CByte

Convierte una expresión a datos de tipo Byte.

CCur

Convierte una expresión a datos de tipo Currency.

CDate

Convierte una expresión a datos de tipo Date.

CDbl

Convierte una expresión a datos de tipo Double (doble precisión).

CDec

Convierte una expresión a datos de tipo Decimal.

CInt

Convierte una expresión a datos de tipo Integer (nombre entero).

CLng

Convierte una expresión a datos de tipo Long (entero largo).

CSng

Convierte una expresión a datos de tipo Single (simple precisión).

© Editions ENI - All rights reserved

Copyrighted Material

Hidden page

QBColor

Devuelve un valor que indica el código de color RGB correspondiente al número de color indicado.

RGB

Devuelve un número entero que representa el valor de un color RGB.

Str, Str$

Devuelve una cadena de caracteres que representa el número especificado.

StrConv

Devuelve un valor convertido al formato indicado.

Val

Devuelve el valor numérico contenido en una cadena de caracteres.

2. Cadenas de caracteres

Asc

Devuelve el código ASCII correspondiente al primer carácter de una cadena.

Chr, Chr$

Devuelve el carácter correspondiente al código ASCII especificado.

InStr

Devuelve la posición de la primera ocurrencia de una cadena dentro de otra cadena.

InStrRev

Devuelve la posición de la ocurrencia de una cadena dentro de otra, a partir del fin de la cadena.

LCase, LCase$

Devuelve una cadena con sus caracteres pasados a minúsculas.

© Editions ENI - All rights reserved

Copyrighted material

Hidden page

Hidden page

Hidden page

Round

Devuelve un número redondeado a una cantidad especificada de posiciones decimales.

Sgn

Devuelve un número entero que indica el signo del argumento.

Sin

Devuelve el seno de un ángulo.

Sqr

Devuelve la raíz cuadrada de un número.

Tan

Devuelve la tangente de un ángulo.

Ejemplo

```
Sub FctsCalculos()
   'Diferencia entre Int y Fix
   nb1 = -125.45
   'Muestra -126
   MsgBox Int(nb1)
   'Muestra -125
   MsgBox Fix(nb1)
   'Devuelve un número aleatorio comprendido entre 1 y 49
   nb2 = Int(49*Rnd)+1
   MsgBox nb2
End Sub
```

Otras funciones, si bien no son intrínsecas, pueden obtenerse a partir de funciones intrínsecas.

Algunos ejemplos:
Secante = 1 / Cos(X).
Cosecante = 1 / Sin(X).
Cotangente = 1 / Tan(X).

© Editions ENI - All rights reserved

Copyrighted Material

Hidden page

Hidden page

DateSerial

Devuelve la fecha correspondiente a una año, un mes y un día especificados.

DateValue

Devuelve una fecha.

Day

Devuelve un número entero comprendido entre 1 y 31 que representa el día del mes.

Hour

Devuelve un número entero comprendido entre 0 y 23 que representa la hora del día.

Minute

Devuelve un número entero comprendido entre 0 y 59 que representa los minutos.

Month

Devuelve un número entero comprendido entre 1 y 12 que representa el mes del año.

MonthName

Devuelve una cadena que indica el mes especificado.

Now

Devuelve la fecha y la hora actuales tomadas del reloj del sistema.

Second

Devuelve un número entero comprendido entre 0 y 59 que representa los segundos.

Time, Time$

Devuelve la hora actual.

Copyrighted material

Timer

Devuelve la cantidad de segundos transcurridos desde la medianoche.

TimeSerial

Devuelve una fecha que contiene la hora exacta (horas, minutos y segundos).

TimeValue

Devuelve una hora.

WeekDay

Devuelve un número entero que representa el día de la semana.

WeekdayName

Devuelve una cadena que indica el día de la semana especificada.

Year

Devuelve un número entero que representa el año.

Ejemplo

Diversos cálculos con fechas y horas:

```
Sub CalcFechasyHoras ()
   '   Muestra la fecha del día
   MsgBox "Hoy es " & Date
   '   Muestra la cantidad de segundos transcurridos desde medianoche
   MsgBox "Medianoche fue hace " & _
          Timer & " segundos"
   '   Calcula y muestra el tiempo que resta trabajar,
   '   suponiendo que la jornada termina a las 17 h 30
   Resto = TimeSerial(17 - Hour(Time), _
           30 - Minute(Time), 0 - Second(Time))
   MsgBox "Finalizando a las 17h30, falta " & Resto & _
          " horas que hay que trabajar"
   '   Calcula y muestra el último día del mes en curso
   Final = DateSerial(Year(Now), Month(Now) + 1, 1) - 1
   MsgBox "el último día del mes en curso es " & Final
   '   Muestra el nombre del día de la semana de esa fecha
   '   (- 1 porque para Excel la semana empieza el domingo)
   MsgBox "Será un " & WeekdayName(Weekday(Final) - 1)
End Sub
```

© Editions ENI - All rights reserved

Copyrighted material

Hidden page

LOF

Devuelve la longitud en bytes de un archivo abierto con la instrucción **Open**.

Seek, Seek$

Devuelve la posición de lectura y escritura actuales en un archivo abierto con la instrucción **Open**.

Ejemplo

Procedimiento para mostrar los nombres, las fechas de última modificación y los tamaños de los cinco primeros archivos encontrados en la carpeta actual.

```
Sub ListaArchivos ()
   Dim strPath As String
   Dim strFile As String

   strPath = CurDir() & "\"
   strFile = Dir(strPath)
   For i = 1 To 5
       If i = 1 Then
          strFile = Dir(strPath)
       Else
          strFile = Dir()
       End If
       If strFile <> "" Then
          MsgBox "Archivo : " & strFile & Chr(13) & _
                 "Fecha : " & FileDateTime(strFile) & Chr(13) & _
                 "Tamaño : " & Format(FileLen(strFile), "# ##0")
      End If
   Next i
End Sub
```

7. Verificación de variables

IsArray

Devuelve un valor que indica si una variable es o no una matriz.

IsDate

Devuelve un valor que indica si una expresión puede convertirse a fecha.

© Editions ENI - All rights reserved

Copyrighted material

Hidden page

Hidden page

Hidden page

Hidden page

Tab

Función utilizada con la instrucción `Print #` o el método `Print` para posicionar la salida.

12.Solver

SolverAdd

Agrega una restricción al problema actual.

SolverChange

Modifica una restricción al problema actual.

SolverDelete

Elimina una restricción al problema actual.

SolverFinish

Indica a Excel qué debe hacer con los resultados y qué clase de informe debe crear al finalizar el proceso de resolución.

SolverFinishDialog

Es igual a la función **SolverFinish**, pero también muestra el cuadro de diálogo **Resultados de Solver** después de resolver el problema.

SolverGet

Devuelve la información relativa a la configuración de Solver.

SolverLoad

Carga la configuración de un modelo existente.

SolverOK

Define un modelo básico de Solver.

SolverOKDialog

Es igual a **SolverOK** pero también muestra el cuadro de diálogo **Solver**.

Copyrighted material

SolverOptions

Especifica las opciones avanzadas de un modelo.

SolverReset

Reinicia toda la configuración.

SolverSave

Guarda la configuración de un modelo.

SolverSolve

Procede con la resolución de un modelo.

C. Constantes VBA

Visual Basic para Aplicaciones permite definir constantes para mejorar la legibilidad del código y facilitar su mantenimiento.

También puede utilizar las siguientes constantes VBA (constantes intrínsecas) en todo su código.

1. Constantes de color

Constante	Valor	Descripción
vbBlack	0x0	Negro
vbRed	0xFF	Rojo
vbGreen	0xFF00	Verde
vbYellow	0xFFFF	Amarillo
vbBlue	0x0FF0000	Azul
vbMagenta	0x0FF00FF	Magenta
vbCyan	0x0FFFF00	Cian
vbWhite	0x0FFFFFF	Blanco

© Editions ENI - All rights reserved

Copyrighted Material

2. Constantes de fecha

Constante	Valor	Descripción
vbSunday	1	Domingo
vbMonday	2	Lunes
vbTuesday	3	Martes
vbWednesday	4	Miércoles
vbThursday	5	Jueves
vbFriday	6	Viernes
vbSaturday	7	Sábado

3. Constantes de teclas correspondientes a letras y números

Los valores de las teclas A a Z son los mismos que sus equivalentes ASCII.

Constante	Valor	Descripción
vbKeyA	65	Tecla A
vbKeyB	66	Tecla B
vbKeyC	67	Tecla C
...	...	...
vbKeyZ	90	Tecla Z

Los valores de las teclas 0 a 9 son los mismos que sus equivalentes ASCII.

Constante	Valor	Descripción
vbKey0	48	Tecla 1
vbKey1	49	Tecla 2
vbKey2	50	Tecla 3
...	...	...
vbKey0	57	Tecla 9

Copyrighted Material

Hidden page

Constante	Valor	Descripción
vbKeyRight	0x27	Tecla [Flecha derecha]
vbKeyDown	0x28	Tecla [Flecha abajo]
vbKeySelect	0x29	Tecla Selección
vbKeyPrint	0x2A	Tecla [Impr Pant]
vbKeyExecute	0x2B	Tecla [EJECUTAR]
vbKeySnapshot	0x2C	Tecla SNAPSHOT
vbKeyInsert	0x2D	Tecla [Insertar]
vbKeyDelete	0x2E	Tecla [Suprimir]
vbKeyHelp	0x2F	Tecla Ayuda
vbKeyNumlock	0x90	Tecla [Bloq Num]
vbKeyLButton	0x1	Botón izquierdo del ratón
vbKeyRButton	0x2	Botón secundario del ratón

Copyrighted material

A

Access
    pilotear desde Excel, *283*
AddIns - colección, *89*
AnswerWizard - objeto, *88*
API, *326*
    ejemplos de uso, *329*
    lista de funciones API Windows, *328*
    llamar a una función de la API Windows, *327*
Application - objeto, *88*
    Véase también Objeto Application
Areas, *161*
Argumentos con nombre, *46*
Assistant - objeto, *88*
AutoCorrect - objeto, *88*
Automation, *273*
    función CreateObject, *274*
    función GetObject, *274*
AutoRecover - objeto, *88*

B

Barra de herramientas de acceso rápido
    agregar un comando, *220*
Barra de herramientas Estándar, *28*
Barras de comandos, *221*
    controles, *221*
    crear, *222*
    eliminar, *223*
Barras de herramientas personalizadas, *220*
Biblioteca de objetos, *103*, *273*

Copyright material

C

Carácter de continuación, *82*
CellFormat - objeto, *88*
Cells - colección, *147*
Chart - objeto, *248*
    Véase también Objeto Chart
Charts - colección, *132*
Ciclos
    Véase Estructuras en ciclo
Clases, *86*
Código VBA, *10*
    entorno Visual Basic Editor, *10*
    grabación de macros, *10*
Colección, *86*, *91*, *95*
    AddIns, *89*
    Charts, *132*
    COMAddIns, *89*
    CommandBars, *89*
    Comments, *148*
    CustomDocumentProperties, *132*
    CustomViews, *132*
    Dialogs, *90*
    Hyperlinks, *148*
    Names, *90*, *132*
    ODBCErrors, *90*
    OLEDBErrors, *90*
    OLEObjects, *294*
    RecentFiles, *90*
    SmartTagRecognizers, *90*
    SmartTags, *148*
    UsedObjects, *90*
    Watches, *90*
    Windows, *90*, *133*
    Workbooks, *90*
    WorkSheets, *90*, *133*

Copyrighted Material

Colección Charts
evento, *247*
COMAddIns - colección, *89*
Comentarios, *82*
CommandBars - colección, *89*
Comments - colección, *148*
Condiciones
Véase también Estructuras de decisión
If... Then, *67*
If... Then... Else... End If, *68*
If... Then... ElseIf..., *69*
If... Then... End If, *67*
Select Case, *70*
Constantes, *63*, *83*, *103*
de color, *383*
de fecha, *384*
de teclas, *384*
de teclas de función, *385*
de teclas diversas, *385*
integradas, *64*
personalizadas, *63*
Cuadros de diálogo
formularios, *180*
integrados, *180*
predefinidos, *180*, *184*
CustomDocumentProperties - colección, *132*
CustomViews - colección, *132*

D

DDE, *290*
clausura, *293*
inicialización, *291*
vinculación, *291*
Debug - objeto, *88*, *264*
DefaultWebOptions - objeto, *88*, *315*

Copyright material

Depuración, *263*
    Véase también Errores
Dialogs - colección, *90*
Do...Loop
    instrucción, *72*

E

Editor VBA, *25*, *30*
    acople de una ventana, *32*
    ingreso de código, *30*
    tipografía, *30*
ErrorCheckingOptions - objeto, *89*
Errores, *260*
    administración, *265*
    de compilación, *261*
    de ejecución, *262*
    de sintaxis, *260*
    depuración, *263*
    Error (función), *268*
    Error (instrucción), *268*
    On Error, *265*
    opciones de recuperación, *31*
Estructuras de decisión, *66*
    condición, *67*
    instrucción IF, *67*
    instrucción Select Case, *70*
Estructuras en ciclo, *72*
    instrucción Do...Loop, *72*
    instrucción For Each...Next, *76*
    instrucción For...Next, *74*
    instrucción While...Wend, *74*
Eventos, *86*, *94*, *103*, *204*, *242*
    cancelar, *208*
Examinador de objetos, *102*
Exit Do, *77*

Copyrighted material

Exit For, *77*
Explorador de objetos, *29*
Explorador de proyectos, *28*

F

Ficha Programador
    descripción, *11*
    mostrar en la cinta de opciones, *11*
FileSystemObject - objeto, *332*
For Each...Next
    instrucción, *76*, *99*
For...Next
    instrucción, *74*
Formularios, *9*, *38*, *40*, *194*
    cierre, *209*
    dimensionar, *195*
    ejecución, *209*
    ejemplo, *211*
    eventos, *204*
    procedimientos, *204*
Función, *10*
    en una formula Excel, *48*
Funciones, *43*
    cadenas de caracteres, *367*
    conversiones, *365*
    exemplos, *50*
    fechas y horas, *373*
    financieras, *372*
    interacción, *378*
    lista, *365*
    matemáticas, *370*
    matrices, *379*
    sistema, *376*
    solver, *382*
    SQL, *380*

Copyrighted Material

Hidden page

Internet
consultas, *303*

L

LanguageSettings - objeto, *89*
Lenguaje VBA
Véase VBA

M

Macros, *11*
asignación, *21*
definir el lugar de almacenamiento, *15*
descripción de las diferentes opciones de seguridad, *18*
ejecutar, *13*
eliminar, *16*
firmas digitales, *19*
grabación, *12*
grabación de macros, *10*
grabar con referencias relativas, *14*
grabar un libro con macros, *16*
las macros y la seguridad, *18*
modificación de los parámetros de seguridad, *18*
modificar, *20*
Matrices, *61*
Mensaje, *186*
Menús, *9*
Barras de comandos, Barras de menú, *9*
Métodos, *86*, *93*, *103*
Microsoft Visual Basic
hojas UserForm, *194*
Modelo de objetos, *87*
Módulos, *10*, *38* - *39*

Copyrighted Material

dar nombre, *40*
eliminar, *40*
exportar, *40*
importar, *40*
pasar, *39*
Módulos de clase, *38*, *40*
MsgBox, *186*

N

Names - colección, *90*, *132*

O

Objeto, *91*, *99*
AnswerWizard, *88*
Assistant, *88*
AutoCorrect, *88*
AutoRecover, *88*
CellFormat, *88*
Debug, *88*
DefaultWebOptions, *88*, *315*
ejemplos de uso, *174*
Err, *268*
ErrorCheckingOptions, *89*
FileSystemObject, *332*
Hyperlink, *323*
LanguageSettings, *89*
OLEObject, *294*
PageSetup, *147*
Protection, *147*
PublishObject, *313*
QueryTable, *305*
SmartTagOptions, *131*

Copyrighted material

Hidden page

de concatenación, *80*
lógicos, *79*
prioridad, *80*
Option Explicit, *57*
Outlook
pilotear desde Excel, *288*

P

PageSetup - objeto, *147*
Procedimiento, *10*
Procedimientos, *40*
acceso, *41*
alcance, *44*
argumentos, *45*
argumentos con nombres, *46*
asociado a eventos, *42*
declaración, *44*
ejecutar, *42*
ejemplos, *50*
eliminar, *42*
function, *41*
general, *42*
llamar, *47*
nombres, *83*
property, *41*
seleccionar, *41*
Sub, *41*
Property, *41*
Propiedades, *86*, *91*, *103*
relativas a la ventana de la aplicación, *119*
Protection - objeto, *147*, *155*
Proyecto, *10*
Publicación de páginas Web, *312*

Copyrighted material

Q

QueryTable - objeto, *305*

R

Range - objeto, *147*, *159*
    Véase también Objeto Range
RecentFiles - colección, *90*
Redacción automática de instrucciones, *97*
Reglas de escritura, *82*
Repetitivas
    Véase Estructuras en ciclo

S

Sangrías, *82*
Seguridad
    descripción de las diferentes opciones de seguridad, *18*
    firmas digitales, *19*
    las macros y la seguridad, *18*
    modificación de los parámetros de seguridad, *18*
Select Case, *70*
Servidor OLE, *273*
Set
    instrucción, *101*
SmartTagOptions - objeto, *131*
SmartTagRecognizers - colección, *90*
SmartTags - colección, *148*
Speech - objeto, *89*
SpellingOptions - objeto, *89*
Sub, *42*
    procedimientos, *41* - *42*

Copyrighted material

T

ThisWorkbook, *38*

U

UsedObjects - colección, *90*

V

Variables, *51*, *83*
- declaración, *57*
- matrices, *61*
- Option Explicit, *57*
- tipos, *52*

VBA, *9*
VBE
- acceso al entorno VBE, *25*
- cerrar el entorno VBE, *26*
- descripción del entorno VBE, *27*
- entorno de desarrollo VBE, *25*
- volver a Excel, *26*

VBE - objeto, *10*, *89*
VBProject, *132*
Ventana Código, *29*
Ventana Inmediato, *29*
Ventana Inspección, *29*
Ventana Locales, *29*
Ventana Propiedades, *28*
Vínculos Excel, *296*
Visual Basic Editor
- *Véase VBE*

Copyrighted material

W

Watches - colección, *90*
WebOptions - objeto, *132*, *315*
While...Wend
    instrucción, *74*
Windows - colección, *90*, *133*
With
    instrucción, *99*
Word
    objetos y colecciones, *277*
    pilotear desde Excel, *276*
Workbook - objeto, *296*
    Véase también Objeto Workbook
Workbooks - colección, *90*
Worksheet
    Véase Objeto Worksheet
WorkSheetFunction - objeto, *89*
WorkSheets - colección, *90*, *133*

X

XML
    importación, exportación y asignación de archivos, *319*
    Modelo Excel XML, *320*

Copyrighted material

Copyrighted material

CONSULTE
nuestro catálogo en internet con:

- la lista de títulos por colección
- una presentación de todos los libros con posibilidad de imprimir el índice
- las novedades editadas cada mes
- la lista de los puntos de venta

www.ediciones-eni.com

"Aprender y dominar la informática"

Pº Ferrocarriles Catalanes, 97-117, 2ª pl. of. 18
08940 - Cornellà de Llobregat (Barcelona)
Tel.: 934 246 401 - Fax: 934 231 576
E-mail: info@ediciones-eni.com

eni ediciones

Copyrighted material

Colección

Recursos Informáticos

Recursos Informáticos

Colección

Redes, sistemas, programación, SGBDR, tecnologías Internet o intranet, informática industrial... Los ámbitos que abarca la informática técnica siguen la evolución de las tecnologías, siendo cada día más variados y complejos. Esta evolución amplía los campos de aprendizaje de los nuevos expertos e impone a los profesionales una continua adaptación de sus conocimientos.

Redactados por formadores profesionales y consultores, los libros de la colección Recursos informáticos le aportan una ayuda eficaz para descubrir y aplicar diferentes técnicas. La presentación de los conceptos básicos, de numerosos ejemplos y los consejos de expertos garantizan la calidad y riqueza de cada uno de estos libros.

VBA Excel 2007

Programar en Excel: Macros y Lenguaje VBA

Completo y, a la vez, simple y práctico, este libro está dirigido a usuarios de Excel o a programadores que deseen crear aplicaciones en plantillas amigables, confiables y poderosas. Otros elementos básicos del lenguaje VBA (estructura del lenguaje y conceptos de programación orientada a objetos) permiten automatizar el tratamiento de sus datos. Usted aprenderá a crear formularios, a personalizar la interfaz de Excel, a comunicar Excel con las otras aplicaciones del paquete Office, a importar y publicar páginas Web y archivos XML y a emplear al máximo las funciones API de Windows. En cada capítulo se presentan numerosos ejemplos. En el último capítulo se desarrolla un ejercicio integrador con la creación de una completa aplicación Excel.
Los ejemplos presentados en el libro pueden descargarse desde el sitio de Ediciones ENI.

Michèle AMELOT

Consultora e formadora en ofimática, Michèle Amelot es una especialista en la creación de programas basados en la tecnología Microsoft. Asesora a numerosos equipos de desarrollo en la dirección de sus proyectos. Es este profundo conocimiento de las necesidades de los usuarios lo que le permite, a través de sus libros, facilitar el aprendizaje y la práctica en la programación en lenguaje VBA.

ISSN 1629-7458
ISBN: 978-2-7460-3915-5
9 782746 039155

Precio: 26.95 €

eni ediciones
www.ediciones-eni.com

Copyrighted material